Learning Invariant Representations for Deep Latent Variable Models

Inauguraldissertation
zur
Erlangung der Würde eines Doktors der Philosophie
vorgelegt der
Philosophisch-Naturwissenschaftlichen Fakultät
der Universität Basel

von
Mario Wieser
aus Deutschland.

2020

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät
auf Antrag von

Prof. Dr. Volker Roth, Dissertationsleiter

Prof. Dr. Thomas Vetter, Korreferent

Basel, den 23. Juni 2020

Prof. Dr. Martin Spiess, Dekan

Für meine Oma Resel und meine Opas Bruno und Willi

Abstract

Deep latent variable models introduce a new class of generative models which are able to handle unstructured data and encode non-linear dependencies. Despite their known flexibility, these models are frequently not invariant against target-specific transformations. Therefore, they suffer from model mismatches and are challenging to interpret or control. We employ the concept of symmetry transformations from physics to formally describe these invariances. In this thesis, we investigate how we can model invariances when a symmetry transformation is either known or unknown. As a consequence, we make contributions in the domain of variable compression under side information and generative modelling.

In our first contribution, we investigate the problem where a symmetry transformation is known yet not implicitly learned by the model. Specifically, we consider the task of estimating mutual information in the context of the deep information bottleneck which is not invariant against monotone transformations. To address this limitation, we extend the deep information bottleneck with a copula construction.

In our second contribution, we address the problem of learning target-invariant subspaces for generative models. In this case, the symmetry transformation is unknown and has to be learned from data. We achieve this by formulating a deep information bottleneck with a target and a target-invariant subspace. To ensure invariance, we provide a continuous mutual information regulariser based on adversarial training.

In our last contribution, we introduce an improved method for learning unknown symmetry transformations with cycle-consistency. To do so, we employ the equivalent deep information bottleneck method with a partitioned latent space. However, we ensure target-invariance by utilizing a cycle-consistency loss in the latent space. As a result, we overcome potential convergence issues introduced by adversarial training and are able to deal with mixed data.

In summary, each of our presented models provide an attempt to better control and understand deep latent variables models by learning symmetry transformations. We demonstrated the effectiveness of our contributions with an extensive evaluation on both artificial and real-world experiments.

Acknowledgments

I would like to thank my advisor Prof. Dr. Volker Roth for all his effort and the opportunity to pursue a PhD in his research group. This thesis would not have been possible without his guidance and is the result of his great support, insightful discussions and ideas. During many enlightening discussions, he has shown me all the connections between different machine learning methods and helped me to develop my view on a machine learning, overall.

I am very thankful to Prof. Dr. Thomas Vetter for agreeing to be a co-examiner of my dissertation and all the time and effort he spent on the review. His advice during many committee meetings and coffee breaks enabled me to look at my problems from a different perspective which helped to significantly improve the presented work.

I also had the privilege to be part of two outstanding interdisciplinary research projects which was a rewarding experience and helped me to look beyond the boundaries of machine learning. I am grateful that I could collaborate with Prof. Dr. Huldrych Günthard, Prof. Dr. Karin Metzner, Prof. Dr. Niko Beerenwinkel, Prof. Dr. Roger Kouyos, Dr. Jasmina Bogojeska and all other members of the SystemsX.ch HIV-X project. In addition, I would also like to thank Prof. Dr. O. Anatole von Lilienfeld, Dr. Jimmy Kromann and all remaining members of the NCCR Marvel for engaging in many interesting research discussions and teaching me the basics of chemistry.

In particular, I would like to thank all current and former members of BMDA: Dr. Sebastian Keller, Dr. Sonali Parbhoo, Aleksander Wieczorek, Damian Murezzan, Daniel Hauke, Maxim Samarin, Fabricio Arend Torres, Vitali Nesterov, Monika Nagy-Huber and Dr. Dinu Kaufmann as well as Dr. Adam Kortylewski, Dr. Andreas Morel-Forster, Dr. Clemens Blumer, Dr. Marcel Lüthi, Dr. Ghazi Bouabene, Dana Rahbani, Dennis Madsen, Patrick Kahr and all other members of GRAVIS for insighful discussions, great social events and a lot of fun during my PhD time.

I especially want to thank Dr. Sebastian Keller, Dr. Adam Kortylewski, Dr. Andreas Morel-Forster, Dr. Sonali Parbhoo and Eric Wieser for proof reading and providing valuable feedback on this PhD thesis.

Zum Schluss möchte ich mich bei den wichtigsten Menschen in meinem Leben bedanken, ohne die ich niemals bis zu diesem Punkt in meinem Leben gekommen wäre. Meiner Mama Isolde und meinem Papa Markus, welche mich in allen Situationen unterstützen, stets zu mir halten und mich immer wieder auf den richtigen Weg zurückgeführt haben. Meinem Bruder Eric, für unser tolles Verhältnis und all seine Hilfe besonders im handwerklichen Bereich ☺. Ganz besonders möchte ich mich auch bei meiner Oma Hilda, meiner leider viel zu früh verstorbenen Oma Resel und meinen leider viel zu früh verstorbenen Opas Bruno und Willi sowie meiner Gotti Relindis bedanken, welche mich auf meinem bisherigen Weg immer unterstützt haben.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 General Motivation

In recent years, deep latent variables models became a crucial cornerstone in the development of modern machine learning methods (Goodfellow et al., 2014; Kingma et al., 2014; Kingma & Welling, 2014). In contrast to traditional latent variable models, a deep latent variable model encodes non-linear dependencies and is able deal with unstructured data. These characteristics open the applicability of latent variables models to a huge number of novel tasks, for example in computational chemistry (Gomez-Bombarelli et al., 2018) or computer vision (Lample et al., 2017). Despite their merits, deep latent variable models are usually unable to learn target-specific invariances from data, implicitly. This frequently leads to problems in understanding and controlling deep latent variables models as well as to model mismatches. To overcome this limitations, we want to explicitly define such invariances in the model. On an abstract level, invariance means that something does not change under a specific transformation. From a mathematical perspective, an invariance can be seen as a property of mappings, where such mappings leave a variable unchanged. In the context of deep latent variables models, invariances appear in various contexts: For instance in facial image analysis where a face could be invariant against certain facial properties such as hair colour or glasses (Klys et al., 2018; Lample et al., 2017). A different example is domain adaptation where we transfer common information from one domain to the other to improve the model performance. However, the transferred information (e.g. the content of an image) should be invariant against domain-specific information (e.g. the style of an image) from the transferred domain (Jha et al., 2018; Lee et al., 2018). As a last example, in fairness we want to be invariant against certain properties that might negatively influence the prediction outcome (Louizos et al., 2016).

In this thesis, we formally describe invariances based on the symmetries from physics. In more detail, a symmetry denotes a quantity which is retained after applying a certain class of symmetry transformations g. In order to calculate a symmetry, we learn a function f which is invariant to g and maps an input to the corresponding symmetry. As a consequence, we denote f as an invariant feature extractor in the reminder of this thesis. A prime example for the concept of symmetries is illustrated in Figure 1.1 by focusing on the task of rotation invariance. To this end, we first observe the 3D representation of a specific object m which is in our case molecule. The molecule m is rotated by applying a symmetry transformation g. For any rotation g, we calculate the distance matrix D between the atoms of the rotated molecule $g(m)$ with a function f that serves as the invariant feature extractor. In simple settings both functions g and f denote

a simple transformation which admits a straightforward analytical form. Due to the fact that g induces an invariance class, we obtain the same distance matrix for every rotation g, i.e. $f(m) = f(g(m))$ for any rotation g.

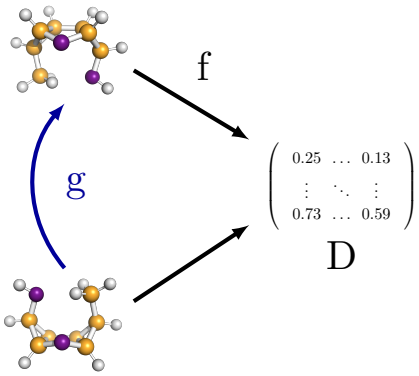


Figure 1.1: A molecule is rotated by g admitting an analytical form. The distance matrix D between atoms is calculated by a known function f and remains unchanged for all rotations.

Symmetry transformations can be considered from different perspectives depending on the task that has to be solved. In this thesis, we assume that the invariant feature extractor f is given analytically but is only described by point-wise samples. Our goal is to find the corresponding symmetry transformation g based on our knowledge about f. In particular, we consider the following tasks where either:

1. g has an analytical form or,
2. g has an unknown form.

In the first task (1), we focus on the estimation of mutual information between two random variables X and Y. In this setting, we have prior knowledge that transforming the marginal distribution of a random variable by applying g will not alter the value of mutual information. However, the function f calculating the mutual information is often complex. A simple calculation is often only feasible for special cases such as the Gaussian distribution. To overcome this limitation, we can approximate f by observing point-wise samples of the random variables X and Y and subsequently learning the parameters of f.

In the second task (2), we investigate the problem where g has unknown form. Here, we can only observe invariances by observing point-wise samples from f while g remains unknown. Consider highly complex domains e.g. the chemical space, where analytical forms of symmetry transformations g are difficult or impossible to find (Figure 1.3). The task of discovering novel molecules for the design of organic solar cells in material science is an example of such a domain. Here, all molecules must possess specific properties, e.g. a bandgap energy of approximately 1.4 eV (Shockley & Queisser, 1961), in order to adequately generate electricity from the solar spectrum. In such scenarios, no predefined symmetry transformation (such as rotation) is known or can be assumed. The only available data defining our invariance class are the $\{m, e\}^n$ numeric point-wise samples from the function f where n is the number of samples, m the molecule and $e = f(m)$ the bandgap energy. Therefore, no analytical form of a symmetry transformation g which alters the molecule m and leaves the bandgap energy e unchanged can be assumed.

1.2 Roadmap and Contribution of the Thesis

After having introduced the concept of symmetries and the overall goal of this thesis, we describe the roadmap and the corresponding contributions in more detail. This thesis is divided into three parts:

In the first part, which is covered by Chapter 4, we focus on learning the function f from point-wise samples while exploiting the fact that the symmetry transformation g is known (see Fig 1.2). In this part, we concentrate on the case where our symmetry is the mutual information and the symmetry transformation g represents the class of all monotone transformations. Our goal is to learn the function f to estimate the mutual information while being invariant against transformations in g. We apply our approach to the Deep Information Bottleneck (Alemi et al., 2017) which is an information-theoretic compression technique. This method compresses a variable X into a random variable Z that retains only information with respect to a target-variable Y. In its traditional form, the deep information bottleneck is not invariant against monotone transformations although it is required by the definition of mutual information. To become invariant against all monotone transformations, we transform the marginals of the random variables with a copula transformation. Subsequently, we are able to estimate the mutual information based on the transformed variables.

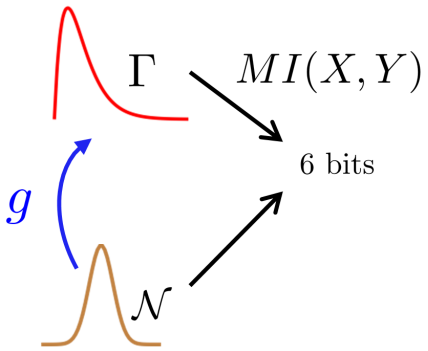


Figure 1.2: This figure illustrates the modeling of symmetries when f and g are known. Here, we focus on the special case of estimating mutual information. Here, f calculates the mutual information between the random variables X and Y whereas g denotes the class of monotone increasing transformations. E.g. the brown part depicts a Gaussian marginal distribution that is transformed to a Beta distribution (red). The mutual information estimate is thus invariant against all transformations of g.

Subsequently, we extend this concept in the second part (Chapter 5) where additionally g is unknown and hence learned from data. The goal of our model is thus to learn the class of symmetry transformations g which result in a symmetry property f of the modelled system. To this end, we learn a continuous data representation and the corresponding symmetry transformation in an inverse fashion from data samples $\{m, e\}_1^n$ only. To do so, we introduce the Symmetry-Transformation Information Bottleneck (STIB). This method encodes the input X (e.g. a molecule) into a latent space Z and subsequently decode it to X and a preselected target property Y (e.g. the bandgap energy). Specifically, we divide the latent space into two subspaces Z_0 and Z_1 to explore the variations of the data with respect to a specific target. Here, Z_1 is the subspace that contains information about input and target, while Z_0 is the subspace that is invariant to the target. In doing so, we capture symmetry transformations not affecting the target Y in the isolated latent space Z_0.

In the last part of this thesis (Chapter 6), we introduce an improved method to learn symmetry transformations by using the concept of cycle-consistency. We employ the equivalent deep information bottleneck method with a partitioned latent space Z_0 and Z_1 as introduced in Chapter 5. In contrast, we do not use a continuous mutual information regulariser in combination with adversarial training. Instead, we ensure the independence assumption between Z_0 and Z_1 by utilizing a cycle-consistency loss. To do so, we fix the Z_0 which encodes the information about Y and sample Z_1 randomly from the prior to generate X. In the next step, we feed the generated X into the encoder and try to map X to the same Z_0. That is, if Z_0 is match, we encode no information about Y in Z_1 thus resulting to independence between Z_0 and Z_1. This approach has various advantages compared to the method in Chapter 5. In the first place, we employ only one minimisation objective to prevent the alternating optimisation process which can lead to convergence problems in practice (Mescheder et al., 2018). Furthermore, our method is able to deal with mixed discrete and continuous Y simultaneously as we do not rely on a continuous or discrete mutual information regulariser.

Despite the fact that we mainly focus on molecular applications it is crucial to note that the developed methods are not limited to chemical problems. On the contrary, these methods are highly universal and may be applied to a broad range of applications areas.

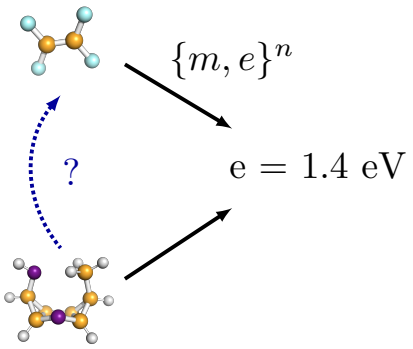


Figure 1.3: n samples $\{m, e\}^n$ where m is the molecule and e the bandgap energy. These samples approximate the function f whereas the class of functions g leading to the same bandgap energy is unknown.

1.3 List of Publications

The following papers have partially resulted from work which is presented in this thesis where stars (*) denote equal contribution. The main thesis is based on the following papers and additional unpublished work:

- *Inverse Learning of Symmetries*, (Wieser et al., 2020)
  Mario Wieser, Sonali Parbhoo, Aleksander Wieczorek, Volker Roth
  Neural Information Processing Systems (NeurIPS), 2020 .

- *Learning Sparse Latent Representation with the Deep Copula Information Bottleneck*, (Wieczorek et al., 2018)
  Aleksander Wieczorek*, Mario Wieser*, Damian Murezzan, Volker Roth
  International Conference on Learning Representations (ICLR), 2018.

In this work, Aleksander Wieczorek contributed equally. Mario Wieser conducted the experiments and contributed to both the writing as well as the model development.

In addition, the subsequent papers have resulted from work during my PhD but are not covered in this thesis:

- *Information Bottleneck For Estimating Treatment Effects with Systematically Missing Covariates*, (Parbhoo et al., 2020)
  Sonali Parbhoo, Mario Wieser, Aleksander Wieczorek, Volker Roth
  Entropy, 2020.

- *Transfer Learning from Well-Curated to Less-Resourced Populations with HIV*, (Parbhoo et al., 2020)
  Sonali Parbhoo, Mario Wieser, Volker Roth, Finale Doshi-Velez
  Machine Learning for Healthcare, 2020.

- *Learning Extremal Representations with Deep Archetypal Analysis*, (Keller et al., 2020)
  Sebastian Keller, Maxim Samarin, Fabricio Arend Torres, Mario Wieser, Volker Roth
  International Journal of Computer Vision (IJCV), 2020.

- *Host genomics of the HIV-1 reservoir size and its decay rate during suppressive antiretroviral treatment*, (Thorball et al., 2020)
  Christian W. Thorball*, Alessandro Borghesi*, Nadine Bachmann, Chantal von Siebenthal, Valentina Vongrad, Teja Turk, Kathrin Neumann, Niko Beerenwinkel, Jasmina Bogojeska, Volker Roth, Yik Lim Kok, Sonali Parbhoo, Mario Wieser, Jürg Boni, Matthieu Perreau, Thomas Klimkait, Sabine Yerly, Manuel Battegay, Andri Rauch, Patrick Schmid, Enos Bernasconi, Matthias Cavassini, Roger D. Kouyos, Huldrych F. Günthard, Karin J. Metzner, Jacques Fellay and the Swiss HIV Cohort Study
  Journal of Acquired Immune Deficiency Syndromes (JAIDS), 2020

- *Intelligent Policy Mixing for Improved HIV-1 Therapy Selection*
  Sonali Parbhoo, Jasmina Bogojeska, Mario Wieser, Fabricio Arend Torres, Maurizio Zazzi, Susana Posada Cespedes, Niko Beerenwinkel, Enos Bernasconi, Manuel Battegay, Alexander Calmy, Matthias Cavassini, Pietro Vernazza, Andri Rauch, Karin J. Metzner, Roger Kouyos, Huldrych Günthard, Finale Doshi-Velez, Volker Roth
  Under review, 2019.

- *Deep Archetypal Analysis*, (Keller et al., 2019)
  Sebastian M. Keller, Maxim Samarin, Mario Wieser, Volker Roth
  German Conference on Pattern Recognition (GCPR), 2019.

- *Determinants of HIV-1 Reservoir Size and Long-Term Dynamics During Suppressive ART*, (Bachmann et al., 2019)
  Nadine Bachmann, Chantal von Siebenthal, Valentina Vongrad, Teja Turk, Kathrin Neumann, Niko Beerenwinkel, Jasmina Bogojeska, Jacques Fellay, Volker Roth, Yik Lim Kok, Christian Thorball, Alessandro Borghesi, Sonali Parbhoo, Mario Wieser, Jürg Boni, Matthieu Perreau, Thomas Klimkait, Sabine Yerly, Manuel Battegay, Andri Rauch, Matthias Hoffmann, Enos Bernasconi, Matthias Cavassini, Roger Kouyos, Karin Metzner, Huldrych Günthard
  Nature Communications, 2019.

- *Greedy Structure Learning of Hierarchical Compositional Models*, (Kortylewski et al., 2019)
  Adam Kortylewski, Aleksander Wieczorek, Mario Wieser, Clemens Blumer, Anreas Morel-Forster, Sonali Parbhoo, Volker Roth and Thomas Vetter
  IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

- *Informed MCMC with Bayesian Neural Networks for Facial Image Analysis*, (Kortylewski et al., 2018)
  Adam Kortylewski*, Mario Wieser*, Andreas Morel-Forster*, Aleksander Wieczorek, Sonali Parbhoo, Volker Roth and Thomas Vetter
  NIPS Bayesian Deep Learning Workshop, 2018.

Chapter 2

Related Work

In this chapter, we review related work on deep latent variable models together with invariant and cycle-consistent representation.

2.1 Deep Latent Variable Models

In this section, we review and discuss related work on the information bottleneck principle, variational autoencoders and copula models.

Information Bottleneck

The information bottleneck principle for discrete random variables was introduced by Tishby et al. (1999). The idea is to compress the random variable X into random variable Z while retaining the information of the random variable Y. This approach can be formalised as the following optimisation problem: $\min_{p(t|x)} I(x;t) - \lambda I(t;y)$. Here, we assume that Y is conditionally independent of Z given X, and where I stands for mutual information and λ for the compression parameter. The multivariate information bottleneck enhance the information bottleneck by allowing for multiple systems of data partitions that are inter-related. In addition, the multivariate information bottleneck has been improved with an agglomerative algorithm (Slonim et al., 2002). Subsequently, the information bottleneck has been augmented from discrete to Gaussian random variables (Chechik et al., 2005) and applied to causaliy (Wieczorek & Roth, 2016).

More recently, research was conducted on deriving variational lower bounds on the information bottleneck optimisation problem (Alemi et al., 2017; Chalk et al., 2016). Both approaches, however, treat the differential entropy of the marginal distribution as a positive constant, which is not always justified as described in Section 5.3). A related model is introduced in Pereyra et al. (2017), where a penalty on the entropy of output distributions of neural networks is imposed.

Variational Autoencoder

The deep information bottleneck has close connections to the variational autoencoder which was introduced by Kingma & Welling (2014) and Rezende et al. (2014). Here, the idea is to combine generative models with variational inference and learn both parts end-to-end. To do so, an encoder is defined which aims to infer the posterior distribution $p(z|x)$ from data. Subsequently, a decoder tries to generate data from the latent representation Z by employing the following conditional distribution $p(x|z)$. However, neural networks are deterministic function why a so-called reparametrisation trick is proposed to introduce stochasticity.

This concept has been refined in various directions. So far, variational autoencoders are merely designed for unsupervised learning. Kingma et al. (2014) augmented this concept to be applicable to semi-supervised learning. Moreover, traditional variational autoencoder require a Gaussian assumption for Z and thus Jang et al. (2017) generalised VAEs to discrete latent Z. Subsequently, Figurnov et al. (2018) proposed a novel reparametristion trick which permitted the usage of additional distributions of Z.

Copula Models

Copulas are heavily employed in finance and statistics because they allow to decouple the dependecy structures from the marginals of multivariate distributions. In machine learning, copulas have gained more and more attention to flexibilise traditional machine learning approaches.

In terms of information theory, copula models were combined with the information bottleneck principle. In (Rey & Roth, 2012), the authors generalised the Gaussian information bottleneck to Meta-Gaussian random variables. The idea is to decouple the marginals and dependency structure where the marginals may be arbitrary distributions and the dependency structure is modelled as a Gaussian copula. Subsequently, this model has been extended to the sparse meta-Gaussian information bottleneck (Rey et al., 2014). Here, a sparsity penalty term was proposed to select features with respect to a specific target variable.

In addition, copula models have been proposed in the context of probabilistic methods. Kaufmann et al. (2015) introduced a copula construction in order to relax the Gaussian assumption in the context of archetypal analysis. Moreover, Suh & Choi (2016) proposed a variational autoencoder model that is augmented with a Gaussian copula. This allows a variational autoencoder to overcome the Gaussian assumption and deal with both discrete and mixed data.

Last, copula methods have been employed to improve shortcomings in terms of variational inference. The most common concept is mean-field variational inference where the idea is to approximate a posterior distribution by assuming that the different random variables of the posterior are independent. Since this is a strong assumption, Tran et al. (2015) introduced a method based on vine-copulas to model the dependencies between the different random variables. Later this concept was refined by Han et al. (2016) who build the dependency structure on Gaussian copulas.

2.2 Invariant Representations

In this section, we discuss related work on invariant representations and draw connections to fairness in machine learning.

Enforcing invariance in latent representations

Bouchacourt et al. introduced a multi-level VAE. Here, the latent space is decomposed into a local feature space that is only relevant for a subgroup and a global feature space. A more common technique to introduce invariance makes use of adversarial networks (Goodfellow et al., 2014). Specifically, the idea is to combine VAEs and GANs, where the discriminator tries to predict attributes, and the encoder network tries to prevent this (Creswell et al., 2017; Lample et al., 2017). Perhaps most closely related to the work presented in this thesis is the approach of Klys et al. (2018) where the authors propose a mutual information regulariser to learn isolated subspaces for binary targets. However, these approaches are only applicable for discrete attributes and our

work tackles the more fundamental and challenging problem of learning symmetry transformations for continuous properties.

Relations to Fairness

In addition, our work has close connections to fairness. The main idea is to penalise the model for presence of nuisance factors S that have an unintended influence on the prediction Y to archive better predictions. Louzios *et al.* (Louizos et al., 2016), for example, developed a fairness constraint for the latent space based on maximum mean discrepancy (MMD) to become invariant to nuisance variables. Later, Xie *et al.* (Xie et al., 2017) proposed an adversarial approach to become invariant against nuisance factors S. In addition, Moyer *et al.* (Moyer et al., 2018) introduced a novel objective to overcome the disadvantages of adversarial training. Subsequently, Jaiswal *et al.* (Jaiswal et al., 2020) built on these methods by introducing a regularisation scheme based on the LSTM Hochreiter & Schmidhuber (1997) forget mechanism. In contrast to the described ideas, our work focuses on learning a symmetry transformation for continuous Y instead of removing nuisance factors S. Furthermore, we are interested in learning a generative model instead of solely improving downstream predictions.

2.3 Cycle-Consistent Representations

Here, we describe related work on cycle-consistent representations and their application in unsupervised image-to-image translation, disentanglement and domain adaptation.

Unsupervised Image-To-Image Translation

The concept of cycle-consistency has been introduced in the setting of unsupervised image-to-image translation (CycleGAN) (Zhu et al., 2017). In this setting, there exist images from two different domains A and B without any direct correspondence between the images. The goal is to translate an image from domain A to domain B in an unsupervised fashion. To do so, a cycle-consistency loss function is proposed which works as follows: an image a is translated from domain A to domain B. Subsequently, the translated image b from domain B is translated to the image a' in domain A. The cycle-consistent loss tries now to minimise the distance between the original image a and the translated image a'. This concept has been extended to unsupervised image-to-image translation tasks in the context of variational autoencoders (Liu et al., 2017) instead of Generative Adversarial Networks (Goodfellow et al., 2014). Here, the author try to map the original image a and the translated image a' on the same point in the latent space by minimising the cycle consistency loss. This concept has further been leveraged for multimodal image-to-image translation (Zhu et al.) as well as few-shot learning (Liu et al., 2019).

Disentanglement and Domain Adaptation

Another important line of research employs the idea of cycle-consistency for learning disentangled representation learning. Jha et al. (2018) proposed a variational autoencoder model to disentangle different factors of variation in the data using cycle-consistent representations. Concurrently, Lee et al. (2018) developed a method to disentangle factors of variations using latent subspaces in a combinations with a novel cycle-consistency loss. Furthermore, cycle-consistency has been utilized in the area of domain adaptation (Hoffman et al., 2018). This approach tries to merge the feature- level and pixel-level adaptation of an image into a single architecture. In

the following, this method has been extended by Mathur et al. (2019) to the case where the label spaces in source and target domains are only partially overlapped.

Chapter 3

Theoretical Background

In this chapter, we revisit the theoretical foundations which are necessary to understand this thesis. In the first Section 3.1, we review the preliminaries of probability theory. In the following Section 3.2, we discuss the theoretical foundations of information theoretic quantities. Subsequently, we introduce the concept of copulas in Section 3.3 and linear latent variable models in Section 3.4. Here, we introduce simple mixture models and extend them to more sophisticated approaches. Last, we draw a connection from linear latent variables models to their non-linear counterpart in Section 3.5.

3.1 Probability Theory

In probability theory part, we discuss their basic foundations which are important to follow the subsequent chapters. For a more intuitive explanation of the described concepts, we will use a fair rolling dice as a running example. That is, the probability to draw a number between one to six is $\frac{1}{6}$.

Probability Space.

The most basic concept in probability theory is a probability space. Such a space is characterised as a triplet $(\Omega, \mathscr{F}, P)$. In this setting, Ω is denoted as the sample space that contains all possible outcomes and is defined as a non-empty set. In the rolling dice example, all possible outcomes are that the dice will fall on one of the following number defined in the set $\{1,2,3,4,5,6\}$. Subsequently, $\mathscr{F}$ depicts the event space which is defined over all subsets of Ω where the elements of $\mathscr{F}$ are further described as events. In the discrete case, the event space is the power set of Ω, that is $\mathscr{F} = 2^{\Omega}$. For the rolling dice example, the event space would be $\mathscr{F} = 2^{\{1,2,3,4,5,6\}}$. For the more complicated continuous case, the event space is characterised as a σ-algebra. The last part of a probability space is the probability measure P. Here, P is defined as the following function $P : \mathscr{F} \rightarrow [0,1]$ which maps events to the corresponding probabilities. For the rolling dice, P would map the event $\{4\}$ to the real value $\frac{1}{6}$.

Random Variable.

In the previous paragraph, we described the abstract concept of probability spaces. However in practice, we are more often interested in the concept of random variables. A random variable is a function that maps from a probability space $(\Omega, \mathscr{F}, P)$ to a measurable space $(\Omega', \mathscr{F}')$ and

subsequently to a real number. In this thesis, we denote random variables with a capital letter X while the observations of such a random variable are written as small letters x. More intuitively, a random variable X could be rolling a dice where we do not know the result beforehand. After rolling the dice, X will attain a value from $1, 2, \ldots, 6$, e.g. $x = 4$.

Probability Mass and Density Function.

In order to describe a random variable, we can use the concept of probability mass and density functions. In terms of a discrete random variable $X : \Omega \rightarrow S$, such a random variable can be described as probability mass function (pmf) $f_X(x) : S \rightarrow [0,1]$ where $S \subseteq \mathbb{R}$ is a discrete subset of $\mathbb{R}$. That is, we assign to each value in S a probability such that:

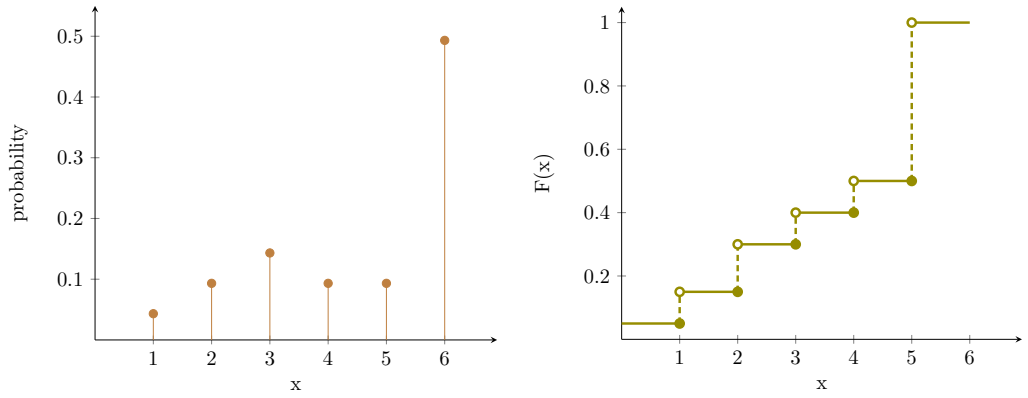
$$f_X(x) = P(X = x), x \in S$$

In the rolling dice example, we may consider the problem of an unfair dice to describe a random variable. That is, it is more likely that this dice will draw a six instead of a one thus we define the pmf such that:

$$p(x) = \begin{cases} 0.05 & x = 1 \\ 0.1 & x = 2 \\ 0.15 & x = 3 \\ 0.1 & x = 4 \\ 0.1 & x = 5 \\ 0.5 & x = 6 \end{cases}$$

A graphical representation of the pmf is illustrated in Figure 3.1a. In contrast, a continuous $X : \Omega \rightarrow \mathbb{R}$ can be formulated as a pmf $f_x(x)$ if there exists a function such that for $x \in \mathbb{R}$:

$$f_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(x)dx$$



(a) (b)

Figure 3.1: A graphical illustration for both a probability mass function (a) and cumulative distribution function (b) for the rolling dice example. In Figure 3.1a, the x-axis depicts the outcome x and the y-axis the corresponding probabilities. In Figure 3.1b, x denote the outcome whereas the y-axis the cumulative distribution function.

Cumulative Distribution Function.

An alternative possibility to define a random variable is a cumulative distribution function (cdf). This is also a prevalent way to characterise a random variable $X : \Omega \to \mathbb{R}$. To do so, a cdf assigns a probability to every value $x \in \mathbb{R}$s a function $F_X(x) : \mathbb{R} \to [0,1]$ which such that:

$$F_X(x) = P(X \leq x)$$

where $P(X \leq x)$ defines the probability that X takes a value that is equal or less than x. In general, a cdf is a non-decreasing and right-continuous function. However in the discrete case, the cdf is discontinuous at points x_i.

As an example, we consider again the problem where the unfair dice is a discrete random variable X. In this setting, we define the cdf as follows:

$$F(x) = \begin{cases} 0 & x < 1 \\ 0.1 & 1 \leq x = 2 \\ 0.15 & 2 \leq x = 3 \\ 0.1 & 3 \leq x = 4 \\ 0.1 & 4 \leq x = 5 \\ 0.5 & 5 \leq x \end{cases}$$

An illustration of the corresponding cdf is shown in Figure 3.1b.

Marginal Distribution.

So far, we have considered merely univariate random variables, that is a random variable which is one-dimensional. In order to define the concept of a marginal distribution, we consider a multivariate $X = (X_1, \ldots, X_n)$ where n are the dimensions and P the probability measure. Hereby, the univariate distributions f_{X_i} of X are called the marginal distribution.

$$f_{X_i}(A) = f_{X_i}(X_1, \ldots, A, \ldots, X_n), A \in \Omega_i$$

In order to calculate the marginal distribution X_i from the joint distribution, we have to sum over all other random variables for discrete cases:

$$f_{X_i} = \sum_{x_1} \cdots \sum_{x_i - 1} \sum_{x_i + 1} \cdots \sum_{x_n} p(X_1, \ldots, X_{i-1}, X_{i+1} \ldots, X_n)$$

Simultaneously, we integrate over all remaining random variables in continuous cases to obtain the marginal distribution X_i:

$$f_{X_i} = \int_{x_1} \cdots \int_{x_i - 1} \int_{x_i + 1} \cdots \int_{x_n} p(X_1, \ldots, X_{i-1}, X_{i+1} \ldots, X_n) \\ dx_1 \ldots dx_{i-1} dx_{i+1} \ldots dx_n$$

For a more intuitive interpretation, we extend the rolling a fair dice example introduced in the previous sections to two dices. For this reason, the random variable X has two dimensions X_1 and X_2 with the joint probability measure P. The probability of receiving a certain combination when

	1	2	3	4	5	6
1	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$
2	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$
3	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$
4	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$
5	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$
6	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$

Table 3.1: This table illustrates the joint probability table for the two dice example. The rows denote the random variable X_1 and the columns X_2.

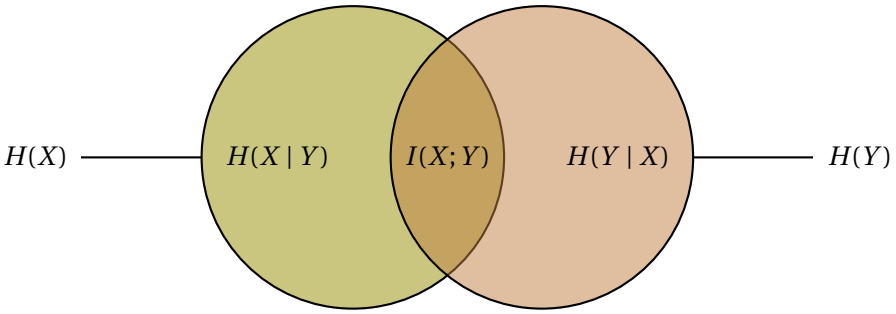


Figure 3.2: In this figure, we illustrate an overview of the most important information theoretic quantities. The green and brown circle denote the entropy $H(X)$ and $H(Y)$ of two random variables X and Y, respectively. The non-overlapping areas of the circles depict the conditional entropies $H(X \mid Y)$ and $H(Y \mid X)$. Finally, the overlapping area of the circles represent the mutual information $I(X;Y)$ between X and Y.

drawing a dice is thus $\frac{1}{36}$ per combination. An illustration of the full example is depicted in Table 3.1 where the rows denote X_1 and the columns X_2.

In order to calculate the marginal distribution of X_1, we have to sum over all columns of X_2:

$$f_{X_1} = \sum_{X_2} p(X_1, X_2) \tag{3.1.1}$$

As a result, we obtain the marginal distribution of X_1 which is $\{\frac{1}{6}\ \frac{1}{6}\ \frac{1}{6}\ \frac{1}{6}\ \frac{1}{6}\ \frac{1}{6}\}$.

3.2 Information Theory

Information Theory was introduced by Shannon and mainly describes the quantification of information in a system. In this section, we only review the most important information theoretic quantities and refer the reader to the work of Cover & Thomas (2006) for a more comprehensive description. An overview of the most important information theoretic concepts is illustrated in Figure 3.2.

3.2.1 Entropy

Entropy denotes the most fundamental quantity within Information Theory where most of the remaining quantities are build on. Here, entropy measures the uncertainty which is in a proba-

bility distribution. That is, how uncertain is the outcome of an event. In other word, what amount of information I is in an event which occurs with probability P. A prime example to explain the concept of entropy is a coin-flip where we distinguish between a fair and a biased coin. By flipping a fair coin, we obtain a high entropy since the probability of getting head or tail is uniform. In contrast, by flipping a biased coin we measure a lower entropy as the probability distribution becomes more deterministic. We start our definition with the Shannon entropy for discrete random variables. To define Shannon entropy, we first need to define the Shannon information content:

Shannon information content:

The Shannon information content of an outcome x is denoted as:

$$h(x) = \log_2 \frac{1}{P(x)}, \tag{3.2.1}$$

That is, the information content is the $\log_2$ of the inverse probability of occurrence. The quantity is measurement unit in bit due to the fact that the Shannon information content is defined as $\log_2$. If we consider the example, we can calculate the Shannon information content for flipping a fair coin. In this setting, the probability of getting head or tail is $\frac{1}{2}$. Therefore, the Shannon information content is 1 bit as we have a maximal uncertainty about the outcome. In contrast, if we consider an unfair coin that shows always head, the Shannon information content is 0 bit as we do not have any uncertainty.

Shannon Entropy:

Based on the Shannon information content, we can define the Shannon entropy of an ensemble X. Here, the main idea is to calculate the average Shannon information content of an outcome. To do so, we define an ensemble as a triple $(x, \mathscr{A}_X, \mathscr{P}_X)$. In this definition, we denote x as the value of a random variable and $\mathscr{A} = a_1, a_2, \ldots, a_N$ as the alphabet of possible values for x. In addition, $\mathscr{P}_X = p_1, p_2, \ldots, p_N$ describes the probabilities for each possible values given in the alphabet $\mathscr{A}_\mathscr{X}$ where $P(x = a_i) = p_i, p_i \geq 0$ and $\sum_{a_i \in \mathscr{A}_\mathscr{X}} P(x = a_i) = 1$:

$$H(X) = \sum_{x \in \mathscr{A}_X} P(x) \log_2 \frac{1}{P(x)} \tag{3.2.2}$$

Thus, the Shannon entropy is defined as the average Shannon information content of an outcome where the following properties hold according to MacKay (2003):

- $H(X) \geq 0$ with equality if and only if $p_i = 1$ for one i.
- $H(X) \leq \log(|\mathscr{A}_X|)$ with equality if and only if $p_i = \frac{1}{|\mathscr{A}_X|}$ for all i, where $|\mathscr{A}_X|$ represent the number of elements in $\mathscr{A}_X$.

In the next step, we are going to calculate the Shannon entropy of an unfair coin (X) flip as an example. The alphabet $\mathscr{A}$ has the possible values heads and tails. The probability $\mathscr{P}$ of each possible value heads and tails is $\frac{1}{4}$ and $\frac{3}{4}$, respectively. To calculate the entropy, we plugin the previously described values in Equation 3.2.2:

$$\begin{aligned} H(X) &= 0.25 \cdot \log_2 \frac{1}{0.25} + 0.75 \cdot \log_2 \frac{1}{0.75} \\ &\approx 0.5 + 0.31 \\ &\approx 0.81 \end{aligned}$$

The entropy of the unfair coin flip is approximately 0.81 bits. Intuitively, this is a reasonable result as we have a lower uncertainty about the outcome of X. In contrast, the entropy of a fair coin would be 1 as the outcome would be completely random.

Differential Entropy:

So far, we have described the Shannon entropy for discrete random variables. However, we can extend the notion of entropy to continuous random variables. To do so, we substitute the sum with an integral which is subsequently called the differential entropy:

$$H(X) = -\int_X f(x) \log_2 f(x) dx, \tag{3.2.3}$$

where f denotes the probability density function. In contrast to Shannon entropy, differential entropy can have negative values.

Conditional Entropy:

The conditional entropy estimates the uncertainty of a random variable X under the assumption that random variable Y is known. We distinguish between the conditional entropy for discrete and continuous random variables. First, we define the conditional entropy for discrete variables:

$$\begin{aligned} H(X \mid Y) &= \sum_{y \in \mathscr{A}_Y} P(y) \left[\sum_{x \in \mathscr{A}_X} P(x \mid y) \log \frac{1}{P(x \mid y)} \right] \\ &= \sum_{xy \in \mathscr{A}_X \mathscr{A}_Y} P(x, y) \log \frac{1}{P(x \mid y)} \end{aligned} \tag{3.2.4}$$

As an example, we extend the unfair coin flip to two coins with two random variables X and Y. Both coins show heads an tail with probability $\frac{1}{4}$ and $\frac{3}{4}$, respectively. To calculate the conditional entropy, we employ Equation 3.2.4. In the first step, we have to sum over all possible outcomes of Y

$$\begin{aligned} H(X \mid Y) = &\frac{1}{4} \cdot \left[\sum_x P(x \mid y) \log \frac{1}{P(x \mid y)} \right] \\ &+ \frac{3}{4} \cdot \left[\sum_x P(x \mid y) \log \frac{1}{P(x \mid y)} \right] \end{aligned}$$

In the second step, we sum over all possible outcomes of X conditioned on Y:

$$
\begin{aligned}
H(X \mid Y) &= \frac{1}{4} \cdot \left[\frac{1}{16} \log_2 \frac{1}{\frac{1}{16}} + \frac{3}{16} \log_2 \frac{1}{\frac{3}{16}} \right] \\
&\quad + \frac{3}{4} \cdot \left[\frac{3}{16} \log_2 \frac{1}{\frac{3}{16}} + \frac{9}{16} \log_2 \frac{1}{\frac{9}{16}} \right] \\
&\approx 0.18 + 0.69 \\
&\approx 0.87
\end{aligned}
$$

Thus, the conditional entropy for our coin flip example is 0.87 bits.

Similarly to the discrete case, we can define conditional entropy for continuous variables with the following form:

$$H(X \mid Y) = \int_{X \times Y} f(x,y) \log \frac{f(x,y)}{f_y(y)} dx, dy \tag{3.2.5}$$

In contrast to the discrete case, we substitute the sum by an integral and the probabilities are replaced by a probability density function f.

3.2.2 Mutual Information

Here, we discuss the idea of mutual information based on the information theoretic quantities introduced in Section 3.2.1. In general, mutual information measures the amount of information about a random variable X while we observe a second random variable Y.

Discrete Case:

Mutual information can be defined in terms of information theoretic quantities or the Kullback-Leibler (KL) divergence. The KL divergence, also known as relative entropy, describes the similarity between two probability distributions $P(x)$ and $Q(x)$. In the discrete case, the KL divergence is defined as:

$$D_{KL}(P \mid\mid Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} \tag{3.2.6}$$

The KL divergence has the important property that it is always non-negative which follows from the Gibbs' inequality which also says that the KL divergence is 0 if and only if $P = Q$:

$$D_{KL}(P \mid\mid Q) \geq 0 \tag{3.2.7}$$

It is important to note that the KL divergence is not symmetric despite the fact that the Kl divergence is often described as KL distance. That is, by switching $P(x)$ and $Q(x)$ we usually do not obtain the same value which in general means that $D_{KL}(P \mid\mid Q) \neq D_{KL}(Q \mid\mid P)$.

As an alternative to the KL divergence, mutual information can also be described as the difference between entropy and conditional entropy:

$$I(X;Y) = H(X) - H(X \mid Y) \tag{3.2.8}$$

Continuous Case:

For continuous random variables the KL divergence is defined over the integral::

$$D_{KL}(P \,||\, Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx \tag{3.2.9}$$

The KL divergence has an analytical solution when both probability distributions P and Q are Gaussian. Here, we assume two multivariate Gaussian distributions P and Q with the same dimension k. The KL divergence can in this case be calculated as:

$$\begin{aligned} D_{KL}(P \,||\, Q) = -\frac{1}{2} \Big[& \mathrm{Tr}(\Sigma_Q^{-1} \Sigma_P) + (\mu_Q - \mu_P)^\top \Sigma_Q^{-1} (\mu_Q - \mu_P) \\ & - k + \ln\left(\frac{\det \Sigma_Q}{\det \Sigma_P}\right) \Big] \end{aligned} \tag{3.2.10}$$

In the above equation, Tr denotes the trace of a matrix and det its determinant. Further, μ_P and μ_Q as well as Σ_P and Σ_Q denote the parameters of the Gaussian distribution P and Q, respectively.

3.3 Copula

Copulas describe a fundamental concept in probability theory and statistics. In simple terms, a copula is a function which couples the marginals and thus allows to specify the dependency structure of a multivariate distribution separately. In the following, we will give a formal definition of copulas and introduce Sklar's theorem.

3.3.1 Definition

For the definition of a Copula C, we consider the following situation where we have n random variables with univariate marginals $F(x_i)$ and $i = 1, \ldots, n$. These n random variables follow the joint distribution $F(x_1, \ldots, x_n)$. In the very simple case where the n random variables are independent from each other, the factorised joint distribution can be factorised such that:

$$F(x_1, \ldots, x_n) = F(x_1) \ldots F(x_n), \tag{3.3.1}$$

by following the well-known laws of probability theory.

In contrast, we may consider a more complex situation where the n random variables share dependencies with each other. In such a situation, the joint distribution cannot be factorised as proposed in Equation 3.3.1. To account for the dependency structure of the joint distribution, we employ the copula C that connects the marginal distributions.

$$F(x_1, \ldots, x_n) = C(F(x_1), \ldots, F(x_n)) \tag{3.3.2}$$

That is, the marginals are independent given a copula C. Subsequently, we give two equivalent definitions of the copula C. First, we define a copula from a probabilistic perspective (Rey, 2015):

Definition 3.3.1. *(Copula.) A d-dimensional copula is a cumulative distribution function* $C : [0,1]^d \rightarrow [0,1]$ *with standard uniform marginal distributions i.e.* $C_j \sim$ *Uniform*$(0,1)$, j.

As an alternative, copulas can also be defined in analytic terms (Rey, 2015):

Definition 3.3.2. *(Copula.) A d-dimensional copula is defined as a function* $C : [0,1]^d \rightarrow [0,1]$ *with the following properties:*

1. $C(u_1,\ldots,u_d)$ *is an increasing function in each component* u_i.
2. $C(u_1,\ldots,u_d) = u_i$ *if* $u_j = 1, j \neq i$ *and* $u_i \in [0,1]$.
3. *For all* $(a_1,\ldots,a_d),(b_1,\ldots,b_d) \in [0,1]^d$ *with* $a_i \leq b_i, \forall i$ *we have:*

$$\sum_{i_1=1}^{2} \cdots \sum_{i_d=1}^{2} (-1)^{i_1 \ldots i_d} C(u_{1,i_1},\ldots,u_{d,i_d}) \geq 0,$$

where $u_{j,1} = a_j$ *and* $u_{j,2} = b_j$.

Next, we introduce the theorem of Sklar (Sklar, 1959) which states the uniqueness and existence of a copula C for a joint distribution $F(X_1,\ldots,_n)$ with corresponding marginals $F(x_i)$ where $i = 1,\ldots,n$. This theorem demonstrates the connection between the copula and multivariate random variables and states the following:

Theorem 3.3.1. *(Sklar.)*

1. *Let* F *be a joint cdf with marginals* $F_1,\ldots,F_d$. *Then there exists a copula* $C : [0,1]^d \rightarrow [0,1]$ *such that*

$$F(x_1,\ldots,x_d) = C(F_1(x_1),\ldots,F_d(x_d)), \forall x_j \in \mathbb{R}. \tag{3.3.3}$$

   A copula is unique, if the marginals are continuous. Apart from that, a copula is uniquely determined on $range(F_1) \times \cdots \times range(F_d)$.

2. *In return,* F *defined as in Eq. 3.3.3 is a multivariate cdf with marginals* $F_1,\ldots,F_d$ *and copula* C, *if* C *is a copula and* $F_1,\ldots,F_d$ *are univariate cdfs.*

That is, we can assure with Sklar's theorem that a valid distribution can be always obtained by a combination of univariate cdfs and a copula.

3.3.2 Probability Integral Transform

However as previously stated, copulas are defined with uniform marginals. To be able to model more complex distributions with copulas, we introduce the concept of the probability integral transform.

$$Y = F_X(X) \tag{3.3.4}$$

The equation states that under the assumption that a random variable X follows a continuous distribution with corresponding cdf F_X, the random variable Y has a standard uniform distribution. That is, any continuous distribution can be transformed to a standard uniform distribution and vice versa with the inverse cdf.

Example: Transforming Uniform to Gaussian

As an example of probability integral transformations, we consider the transformation of a Uniform random variable to a Gaussian random variable. To do so, we uniformly sample 10000 data points. A visualisation is illustrated on the x-axis of Figure 3.3. In order to transform the variable

to Gaussian, we employ the inverse Gaussian cdf which stretches the outer regions of the Uniform distribution which subsequently leads to a Gaussian. The result is depicted on the y-axis of Figure 3.3.

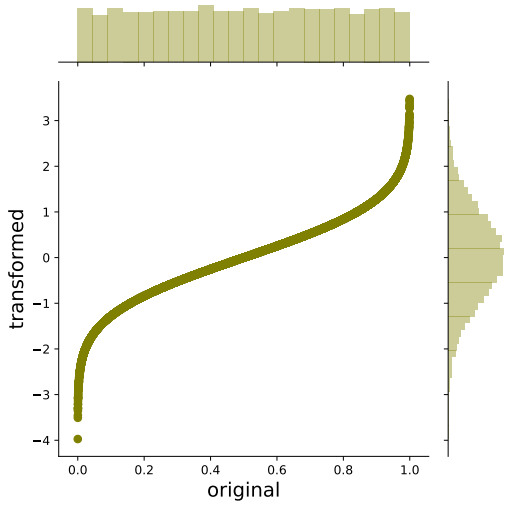


Figure 3.3: This figure illustrates the transformation of a Uniform to a Gaussian random variable. On the x-axis, we have the original random variable with uniform distribution. On the y-axis, we see the transformed variable after applying the inverse Gaussian cdf.

3.3.3 Copula Families

Despite there exist various copula families, we review the two most prevalent ones, namely the independence and Gaussian copula. For a more detailed discussion on other copula families, we refer the reader to Nelsen (2010).

The simplest form of a copula is denoted as the independence copula and defined as follows (Rey, 2015):

Definition 3.3.3. *(Independence Copula.) The independence copula is defined by:*

$$C(u_1,\dots,u_d) = \Pi(u_1,\dots,u_d) = u_1 \dots u_d \tag{3.3.5}$$

A more sophisticated copula is the Gaussian copula which has many application areas such as finance (Rey, 2015).

Definition 3.3.4. *(Gaussian Copula.) The copula C_G of a multivariate Gaussian random variable $X \sim \mathcal{N}(\mu,\Sigma)$ is called Gaussian copula with parameter matrix P, where P is the correlation matrix of X.*

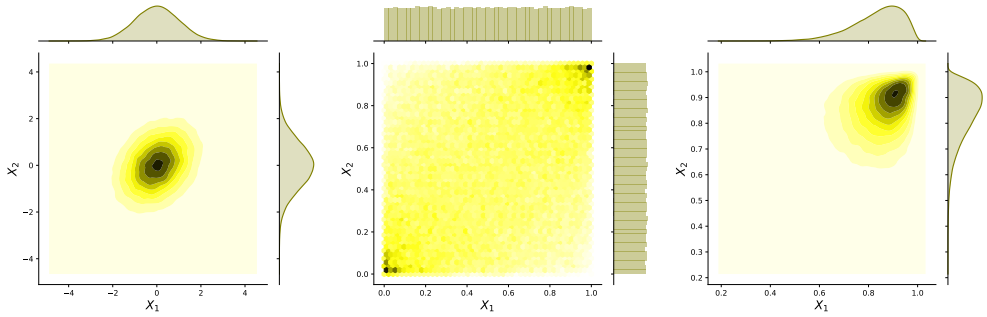
It is interesting to note, that the copula C_G is only dependent of the correlation matrix P yet not the moments μ and σ. That is, all Gaussian variables with the identical correlations matrix P posses the same Gaussian copula C_G independent of their moments μ and σ. This follows from the property that copulas are invariant against strictly increasing transformations.

Example: Modelling Complex Distributions with Gaussian Copulas

In this example, we demonstrate how to model a joint probability distributions with Beta marginals and Gaussian dependency structure by employing a Gaussian copula construction. To do so, we decouple the marginal distributions from the Gaussian dependency structure. Therefore, we first sample 10000 data points from the following a two-dimensional Gaussian distribution X:

$$X \sim \mathcal{N}\left(0, \begin{pmatrix} 1.0 & 0.3 \\ 0.3 & 1.0 \end{pmatrix}\right) \tag{3.3.6}$$

An visual illustration of this distributions is depicted in Figure 3.4a. As mentioned before, a copula consists of uniform marginals coupled by a copula C. Hence, we transform the Gaussian marginal to Uniform by applying the probability integral transform described in Section 3.3.2 which results in a Gaussian copula with Uniform marginals (Figure 3.4b). Subsequently, we transform all Uniform marginal distributions to Beta marginal distributions which is illustrated in Figure 3.4c. That is, we obtained a joint probability distribution with Gaussian dependency structure and Beta marginals.



(a) (b) (c)

Figure 3.4: A graphical illustration for modelling complex joint distributions with Gaussian copulas. First, we start with a Gaussian distribution (Figure 3.4a) and transform the marginals to Uniform to obtain a Gaussian copula (Figure 3.4b). Finally, we transform the Uniform marginals to a Beta distribution which is depicted in Figure 3.4c. This results in a joint probability distribution with Gaussian dependency structure and Beta marginals.

3.4 Linear Latent Variable Models

The idea to model probability distributions on its own is not sufficient to model the complexity of the world. For this reason, we introduce the concept of latent random variables (short latent variables or latent space) variables which allow us to model more complex tasks. In contrast to observed random variables, latent variables cannot directly be observed but have be inferred from other observed random variables. A common reason for employing latent variables is that observed random variables may be influenced by some hidden cause. In addition, this method has the advantage that it is possible to model a specific task with less parameters as we can apply the concept of conditional independence. Moreover, latent variables can serve as a bottleneck, where these random variables are a compact representation of the observed variables. That is, we can express an observed high-dimensional random variable as a low-dimensional hidden random variable which may lead to better interpretability of the data. In the following, we describe

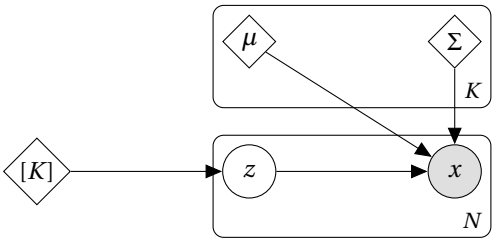


Figure 3.5: illustrates the plate diagram of a Gaussian Mixture Model (GMM). Grey circles indicate observed random variables, whites circles denote latent random variables and diamonds fixed parameters. In addition, rectangles represent the number of repetitions and the square brackets mean a fixed vector of K.

the most common linear latent variables models and further discuss their properties.

3.4.1 Mixture Models

We start with the simplest latent variable model which is also known as mixture model. Here, the idea is to represent the observed random variable X by a mixture of latent random variables $Z_i \in \{1,\dots,K\}$. We determine the latent variables to be discrete for simplicity. To do so, we define a discrete prior $p(z_i) = \text{Cat}(\pi)$ for the latent variables where Cat denotes the Categorical distribution with hyperparameter π. Subsequently, we define the likelihood function of our model as $p(x_i|z_i = k) = p_k(x_i)$ where p_k denotes k'th base distribution. Note, there are no restrictions for choosing the base distribution. Thus in summary, our mixture model may be expressed as:

$$p(x_i|\theta) = \sum_{k=1}^{K} = \pi_k p_k(x_i|\theta) \tag{3.4.1}$$

where π_k denotes mixing weights for our base distributions and θ the model parameters. These mixing weights lead to the property that x may be expressed as convex combinations of the latent variables z. In doing so, π_k has to fullfill the following conditions: Every π_k needs to have a value that lies within the range $0 \leq \pi_k \leq 1$. Furthermore, all π_k's have to sum to 1: $\sum_{k=1}^{K} \pi_k = 1$. The most prominent example of mixture models are Gaussian mixture models:

Example: Mixtures of Gaussians

In a Gaussian Mixture Models (GMM), as the name already explains, we model each base distribution as a Gaussian distribution. A prime example for this approach is soft clustering of data points. Therefore, we predefine the number of clusters as k base distributions and π as the cluster assignment probabilities for each data point. An illustration of the model is depicted in Figure 3.5. More formally, we express our base distribution as a multivariate Gaussian with mean μ_k and covariance matrix Σ_k. Hence, the full model can be expressed as:

$$p(x_i|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mu_k, \Sigma_k) \tag{3.4.2}$$

3.4.2 Factor Analysis

In Section 3.4.1, we introduced the concept of a mixture model where we fixed the latent random variables to be discrete for simplicity. However, this leads to a limited representational power

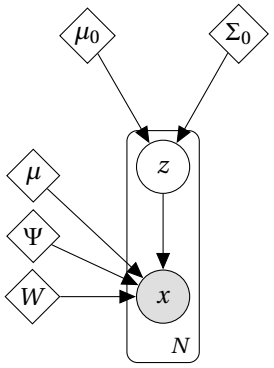


Figure 3.6: This Figure illustrates the plate diagram of a Factor Analysis Model (FA). Here, grey circles denote observed random variables whereas whites circles represent latent random variables. Moreover, diamonds indicate fixed parameters and rectangles represent the number of repetitions.

of our model. For example, in mixture models, our observations can only be generated from one base distribution. In the GMM case, this means that one data point can only belong to one cluster. That is, all latent variables are mutual exclusive. In order to overcome these limitations, we introduce factor models. A graphical illustration is visualised in Figure 3.6. Here, the idea is to relax the restriction that the latent variables are discrete and now allow for continuous latent variables $Z_i \in \mathbb{R}^L$. In general, the prior can take the form of any continuous probability distribution. However, we describe our model in its simplest form and therefore we assume a Gaussian prior for the latent variables Z:

$$p(z) = \mathcal{N}(z \mid \mu_0, \Sigma_0) \tag{3.4.3}$$

where μ_0 and Σ_0 describe the fixed mean and covariance parameters of our prior. As a likelihood, we choose a Gaussian which may be substituted by any other form:

$$p(x \mid z, \theta) = \mathcal{N}(Wz + \mu, \Psi) \tag{3.4.4}$$

The mean is hereby defined as the latent variables Z and a factor loading matrix W which has the following dimensions $D \times L$. This matrix defines the the correlation between the observed and the latent variables. In addition, Ψ is a $D \times D$ covariance matrix which is constrained to be diagonal as we want to explain the correlation.

Example: Principal Component Analysis

A special case of factor analysis is principle component analysis where we try to extract the principal components of the data. This is a common technique for dimensionality reduction as we try to explain the maximal variance of the data by a fixed number of principal components. The main difference to factor analysis is that Ψ is now denoted as $\sigma^2 I$. If $\sigma^2 = 0$, we end up in the classical PCA setting whereas $\sigma^2 > 0$ leads to probabilistic PCA. Given the above described differences, we can express PCA as the following model:

$$p(z) = \mathcal{N}(z \mid \mu_0, \Sigma_0) \tag{3.4.5}$$

where $p(z)$ denotes our prior with mean μ_0 and covariance Σ_0. Subsequently, we can write our likelihood as:

$$p(x \mid z, \theta) = \mathcal{N}(Wz + \mu, \sigma I) \tag{3.4.6}$$

where W is the factor loading matrix and the covariance is changed to σI.

3.4.3 Canonical Correlation Analysis

In real world, there often exist multiple views on the same problem which cannot be modeled with the approaches we have introduced so far. Consider an example from computer vision where we would like to explain the variance of the data. In this case, the first view could be the actual image and the second view a textual description of the image. To deal with such cases, we introduce a multi-view version of PCA which is called canonical correlation analysis (CCA). A graphical representation is illustrated in Figure 3.7. The idea is that each view has its own private subspace that only influences one of the observed random variables X or Y which are later denoted as views. In addition, there exists a third subspace that is shared between the two observed variables X and Y. With this approach, we can model the conical correlations that affect both views X and Y. More formally, we define private subspaces as $Z^x \in \mathbb{R}^{L_x}$ and $z^y \in \mathbb{R}^{L_y}$. The common subspace is defined as $Z^s \in \mathbb{R}^{L_s}$. As previously described in factor analysis, we model our priors as Gaussian distributions:

$$\begin{aligned} p(z^s) &= \mathcal{N}(z^s \mid 0, I_{L_s}) \\ p(z^x) &= \mathcal{N}(z^x \mid 0, I_{L_x}) \\ p(z^y) &= \mathcal{N}(z^y \mid 0, I_{L_y}) \end{aligned} \tag{3.4.7}$$

In contrast to PCA, we set the mean and covariance parameters of all three Gaussian priors to 0 and I, respectively. The likelihoods are extended to the following form. For the first view, we model the likelihood as:

$$p(x \mid z) = \mathcal{N}(x \mid B_x z^x + W_x z^s + \mu_x, \sigma^2 I_{D_x}) \tag{3.4.8}$$

where, we obtain two factor loading matrices. The first factor loading matrix B_x depicts the correlation between X and the private latent variable Z^x whereas W_x the correlation between X and the common space Z^s. The same holds for the second view with the observational variable Y:

$$p(y \mid z) = \mathcal{N}(y \mid B_y z^y + W_y z^s + \mu_y, \sigma^2 I_{D_y}) \tag{3.4.9}$$

Here, we define the factor loading matrix for the private space as Z^y as B_y and for the common space Z^s as W_y.

3.4.4 The Information Bottleneck

To this point, we formulated all linear latent variables models in terms of a Gaussian distribution. However, the usage of a certain distribution imposes often too strong assumptions about the underlying data which may lead to model mismatches. A different approach to define linear latent variables models is to use the concept of information theory where a linear latent variable model is defined with the information bottleneck approach Tishby et al. (1999). This relaxes the assumption to model the data as an explicit distribution. Here, an arbitrary distribution is

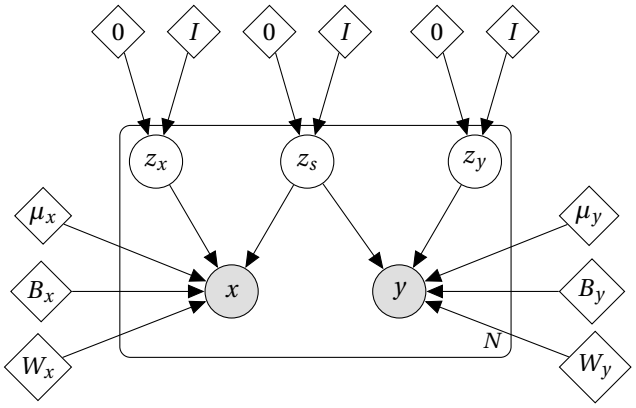


Figure 3.7: illustrates the plate diagram of a Canonical Correlation Analysis (CCA) model. Here, grey circles denote observed random variables whereas whites circles represent latent random variables. Moreover, diamonds indicate fixed parameters and rectangles represent the number of repetitions.

expressed with the information theoretic concept of mutual information. To do so, we define the following optimisation objective;

$$\min I(X;Z) - \lambda I(Z;Y) \tag{3.4.10}$$

The main idea is to compress a variable X into a latent variable Z such that Z only retains information about a third variable Y. The tradeoff between compression and prediction can be adjusted by a compression parameter λ where $\lambda \geq 0$.

However, a drawback of this method is that it is difficult to optimise. For discrete random variables, there exists an algorithm called Blahut-Arimoto which allows to optimise this problem. To overcome the limitation of discrete variables, a special formulation for Gaussian random variables has been found by Chechik et al. (2005), which is closely related to supervised PCA.

Gaussian Information Bottleneck

As previously described, the Information Bottleneck does not have an analytical solution and thus has to be optimised by algorithms such as Blahut-Arimoto. However, there exists an analytical solution for the Information Bottleneck principle in the Gaussian case. Here, we assume that both the input X and the target Y are jointly Gaussian distributed which leads to the Gaussian Information Bottleneck (Chechik et al., 2005). This assumption leads to the fact that also the solution Z of (3.4.10) is also Gaussian distributed.

$$(X,Y) \sim \mathcal{N}\left(0, \begin{pmatrix} \Sigma_X & \Sigma_{XY} \\ \Sigma_{XY}{}^{\top} & \Sigma_Y \end{pmatrix}\right)$$

In more detail, we can describe the compression Z as a noisy linear projection of X:

$$Z = AX + \xi,$$

where $\xi \sim \mathcal{N}(0, \Sigma_\xi)$ is independent of X. That is that:

$$Z \sim \mathcal{N}(0, A\Sigma_X A^T + \Sigma_\xi).$$

This leads to the fact that we can rewrite the Information Bottleneck optimisation problem (3.4.10) as the following optimisation problem:

$$\min_{A,\Sigma_\xi} I(X; AX+\xi) - \lambda I(AX+\xi; Y)$$

As we are in the Gaussian setting, we can formulate the entropy for n-dimensional Gaussian random variables. As a result, we can write the mutual information as follows (Cover & Thomas, 2006):

$$\begin{aligned} I(X;Y) &= h(X) - h(X|Y) \\ &= \frac{1}{2}\log\big((2\pi e)^n|\Sigma_X|\big) - \frac{1}{2}\log\big((2\pi e)^n|\Sigma_{X|Y}|\big) \end{aligned}$$

where Σ_X and $\Sigma_{X|Y}$ denote covariance matrices of X and $X|Y$, respectively. As a consequence of the Gaussian formulation, we obtain an analytical solution for the mutual information.

3.5 Non-Linear Latent Variable Models

In Section 3.4, we described to concept of linear latent variables models. However, these models suffer from the limitation that they can only capture linear dependencies. In this section, we first draw connections between CCA and the information bottleneck and show that the information bottleneck is an asymmetric version of CCA. A graphical scetch is illustrated in Figure 3.8. Subsequently, we introduce a non-linear version the information bottleneck and show the connection to variational autoencoders (VAE).

(a) CCA (b) Information Bottleneck

Figure 3.8: This figure illustrates the connection between CCA and the Information Bottleneck by using a simplified plate diagram. Here, grey circles denote observed random variables whereas whites circles represent latent random variables. Moreover, rectangles represent the number of repetitions.

3.5.1 Deep Information Bottleneck

The Deep Variational Information Bottleneck (DVIB) (Alemi et al., 2017) is a compression technique based on mutual information. We achieve the optimal compression by solving the following parametric problem:

$$\min_{\phi,\theta} I_\phi(X;Z) - \lambda I_{\phi,\theta}(Z;Y), \tag{3.5.1}$$

where we assume a parametric form of the conditionals $p_\phi(z|x)$ and $p_\theta(y|z)$. I denotes the mutual information between two random variables and λ trades-off the degree of compression versus prediction. In addition, the subscripts denote our trainable neural network parameters.

The mutual information terms can be expressed as:

$$
\begin{aligned}
I_\phi(Z;X) = \; & D_{KL}\big(p(z|x)\,p(x)\,\|\,p(z)\,p(x)\big) \\
= \; & \int p(z,x)\log p_\phi(z|x)\mathrm{d}x\,\mathrm{d}t \\
& - \int p(x|z)\,p(z)\log p(z)\mathrm{d}x\,\mathrm{d}t \\
= \; & \int p_\phi(z|x)\,p(x)\log\frac{p_\phi(z|x)}{p(z)}\,\mathrm{d}x\,\mathrm{d}t \\
= \; & \mathbb{E}_{p(x)} D_{KL}\big(p_\phi(z|x)\,\|\,p(z)\big)
\end{aligned}
$$

$$
\begin{aligned}
I_{\phi,\theta}(Z;Y) = \; & D_{KL}\left(\left[\int p(z|y,x)\,p(y,x)\,\mathrm{d}x\right]\|\,p(z)\,p(y)\right) \\
= \; & \int p_\phi(z|x,y)\,p(x,y)\log\frac{p_\theta(y|z)\,p(z)}{p(z)\,p(y)}\,\mathrm{d}z\,\mathrm{d}x\,\mathrm{d}y \\
= \; & \mathbb{E}_{p(x,y)}\left[\int p_\phi(z|x,y)\log p_\theta(y|z)\,\mathrm{d}z\right] \\
& - \mathbb{E}_{p(x,y)}\left[\log p(y)\int p_\phi(z|x,y)\,\mathrm{d}z\right] \\
= \; & \mathbb{E}_{p(x,y)}\mathbb{E}_{p_\phi(z|x)}\log p_\theta(y|z) + h(Y),
\end{aligned}
$$

where the last equality in Eq. (3.5.2) follows from the Markov assumption $Z - X - Y$ in the information bottleneck model: $p_\phi(z|x,y) = p_\phi(z|x)$.

It is important to note that both Markov chains $Z - X - Y$ and $X - Z - Y$ have to hold in the non-linear information bottleneck approach. This leads to the effect that it is limiting for the set of potential solutions. In order to overcome this situation, we are optimising the lower bound for $I_{\phi,\theta}(Z;Y)$. In this particular case, solely the latter Markov chain has to hold (Wieczorek & Roth, 2020). As a result,we formulate $I_{\phi,\theta}(Z;Y)$ as follows:

$$
\begin{aligned}
I(Z;Y) &= \mathbb{E}_{p(x,y)}\int p(z|x,y)\log p(y|z)\,\mathrm{d}t + h(Y) \\
&= \mathbb{E}_{p(x)}\mathbb{E}_{p(y|x)}\int p(z|x,y)\log p(y|zx)\,\mathrm{d}t + h(Y),
\end{aligned}
$$

and which leads to the following bound after so mathematical reformulation:

$$
\begin{aligned}
I(Z;Y) &= \mathbb{E}_{p(x)}\mathbb{E}_{p(y|x)}\mathbb{E}_{p(z|xy)}\log p(y|z) + h(Y) \\
&= \mathbb{E}_{p(x)}\mathbb{E}_{p(y|x)}\mathbb{E}_{p(z|x)}\log p(y|z) \\
&\quad + D_{KL}(p(y,z|x)\,\|\,p(y|x)\,p(z|x)) \\
&\quad + D_{KL}(p(y|x)\,p(z|x)\,\|\,p(y,z|x)) + h(Y) \\
&\geq \mathbb{E}_{p(x)}\mathbb{E}_{p(y|x)}\mathbb{E}_{p(z|x)}\log p(y|z) + h(Y).
\end{aligned}
$$

instead of using $Z - X - Y$ in the last step of Eq. (3.5.2):

Using the reparametrisation trick, the model can be trained using stochastic gradient descent by assuming a simple Gaussian prior $p(z) = \mathcal{N}(z;0,I)$.

3.5.2 Variational Autoencoder

The variational autoencoder (VAE) is a deep generative model that combines the generative process (decoder) with variational inference (encoder)(Kingma & Welling, 2014; Rezende et al., 2014) to learn a probabilistic model. In general, this is a deep information bottleneck model where Y

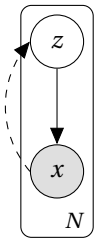


Figure 3.9: This figure denotes the VAE model. Grey circles denote observed random variables whereas whites circles represent latent random variables. The black arrow denotes the decoder part of the VAE whereas the dotted arrow represents the encoder.

is replaced by X. The goal of the encoder is to model the posterior distribution $q(z \mid x)$ where we assume a Gaussian distribution with parameters μ_z and σ_z. In addition, we set $p(z)$ to be an isotropic Gaussian prior $\mathcal{N}(0,1)$.

In more detail, the encoder is expressed as

$$q(z \mid x) = D_{KL}[q(z \mid x) \mid\mid p(z)]$$

where D_{KL} denotes the Kluback-Leibler divergence between the posterior and the prior. The decoder models the generative distribution by $p(x \mid z)$ which defined by the parameters μ_x and σ_x. Thus, the decoder can be denoted as:

$$p(x \mid z) = \mathbb{E}_{z \sim q(z|x)}[p(x \mid z)]$$

In practice, the encoder and decoder are defined as neural networks. Due to the deterministic nature of neural networks, we have to reparametrize the random variable z as $z = \mu_z + \sigma_z * \epsilon$ where $\epsilon \sim \mathcal{N}(0,1)$. To learn the probabilistic model, we optimize the following lower bound. For further information of the derivation, we refer the reader to the work of (Kingma & Welling, 2014; Rezende et al., 2014):

$$p(x) \geq \mathbb{E}_{z \sim q(z|x)}[p(x \mid z)] - D_{KL}[q(z \mid x) \mid\mid p(z)]$$

Chapter 4

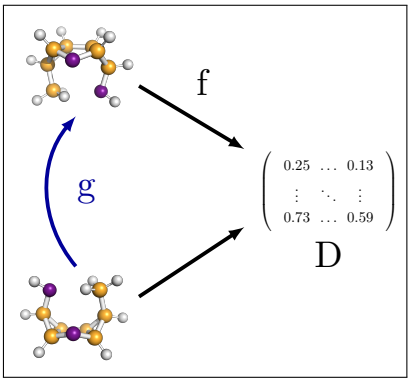
Learning Symmetries by Property Exploitation

Parts of this chapter have already been published at the International Conference on Learning Representations (ICLR). Thus, this work closely follows Wieczorek et al. (2018).

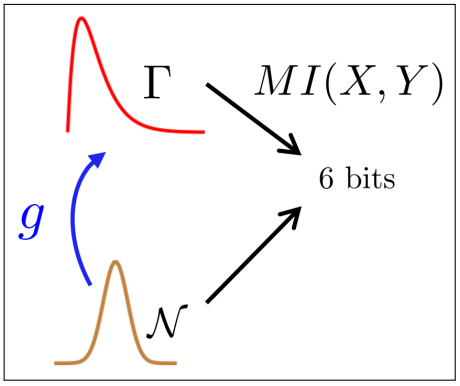
4.1 Introduction

A symmetry denotes a specific property that stays invariant after some transformation. Reconsider the example which was given in the introduction in Chapter 1. A molecule m was rotated by some known symmetry transformation g which lead to the same distance matrix thus denoting our symmetry (see Figure 4.1a). In such simple cases, the function f which calculates out symmetry admits an analytical form. However, in the predominant cases the function f is unknown or difficult to calculate and hence need to be learned from data samples. In this chapter, we focus on the task when the inverse feature extractor f is a mutual information. A graphical sketch of this problem is depicted in Figure 4.1b. Here, we know beforehand the symmetry transformation g which is the class of all monotone transformations that may be applied to the marginals of a distribution. However, estimating mutual information is important yet challenging for many machine learning algorithms such as the information bottleneck (Tishby et al., 1999). Despite the fact that we know the theoretical properties of mutual information there exists no analytical solution expect for special cases such as Gaussian random variables. In this chapter, we focus on estimating mutual information for the information bottleneck principle introduced by (Tishby et al., 1999). The *information bottleneck (IB)* principle identifies relevant features with respect to a target variable. It takes two random vectors X and Y and searches for a third random vector Z which, compresses X and preserves the information contained in Y.

In its traditional form, the information bottleneck is defined for discrete random variables and an approximate solution can be found with the Blahut-Arimoto algorithm for rate-distortion function calculation (Tishby et al., 1999). To overcome the limitation of discrete random variables, the information bottleneck has been extended to continuous Gaussain random variables by Chechik et al. (2005). The reason is that the information bottleneck solution only depends on the copula of X and Y and is thus invariant to strictly monotone transformations of the marginal distributions. However, in various cases the marginal distribution of random variables are not necessarily Gaussian which thus leads to incorrect mutual information estimates. To overcome this issue, the Meta-Gaussian information bottleneck has been developed which is based on a



(a) Rotational symmetry transformation that leads to the same distance matrix.



(b) Monotone increasing symmetry transformation with mutual information f.

Figure 4.1: **Left:** A molecule is rotated by g admitting an analytical form. The distance matrix D between atoms is calculated by a known function f and remains unchanged for all rotations. **Right:** X and Y are exponentially distributed random variables. Our function f calculates the mutual information MI between X and Y. The class of functions g denote monotone increasing transformations which transforms a Gaussian to a Gamma distribution but leads to the same mutual information.

Gaussian copula construction (Rey & Roth, 2012). This construction allows for arbitrary marginal distribution with a Gaussian dependency structure.

However, the methods mentioned in the previous paragraph suffer from the drawback that they only allow for linear compression. For this reason, the Deep Variational Information Bottleneck (DIB), a non-linear version of the information bottleneck has been proposed by Alemi et al. (2017). Here, the authors have shown that variational autoencoders (Kingma & Welling, 2014; Rezende et al., 2014) constitute a special case for the information bottleneck if the compression parameter λ is equal to one. Similar to the Gaussian Information Bottleneck, DIB is defined with Gaussian margins. That is, the mutual information estimate is not invariant against monotone increasing transformations. DIB does not preserve this invariance, which means that DIB fails to implicitly model the marginal distributions. We elaborate on the fundamental issues arising from this lack of invariance in Section 5.3.

The goal of this chapter is to learn the function f which estimates the mutual information while being invariant against monotone increasing transformations g. To this end, we restore the invariance properties of the mutual information by applying a copula transformation to X and Y. This approach makes the mutual information only depend on the copula but not on the marginal distributions. As a consequence, the problems arising from the lack of invariance to monotone transformations of the marginals are solved. By this means, we receive a solution to fully represent all the desirable features inherent to the information bottleneck formulation. The model is therefore simplified by ensuring robust and fully non-parametric treatment of the marginal distributions.

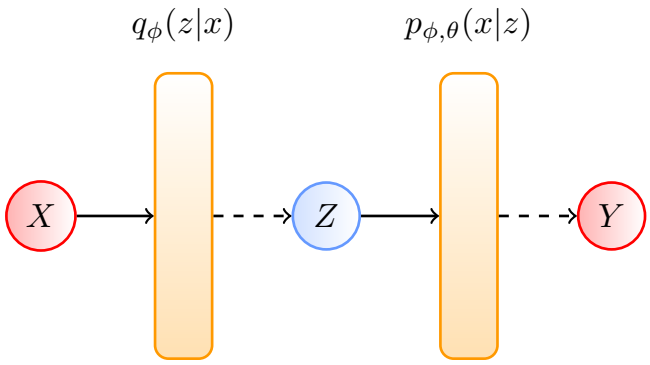


Figure 4.2: This figure illustrates the Deep information bottleneck which is the starting point of our approach. Here, orange rectangles denote neural networks that parametrise the mutual information terms of our model. The blue circle represents a latent random variable whereas the red circle denotes an observed random variable.

4.2 Deep Information Bottleneck

The starting point of our approach is a parametric formulation of the information bottleneck (Alemi et al., 2017) which is specified as follows:

$$\min_{\phi,\theta} I_\phi(X;Z) - \lambda I_{\phi,\theta}(Z;Y), \tag{4.2.1}$$

In this formulation, we depict I as the mutual information between two random variables. The subscripts ϕ and θ are the parameters which are learned during the training process. A graphical illustration of our model is visualised in Figure 4.2.

In more detail, the mutual information terms which are described in Eq. (4.2.1) can be written as follows: The first mutual information term $I_\phi(X;Z)$, that forms the encoder of our model is expressed as:

$$I_\phi(Z;X) = \mathbb{E}_{p(x)} D_{KL}\big(p_\phi(z|x) \| p(z)\big), \tag{4.2.2}$$

where D_{KL} denotes the Kluback-Leibler divergence between the parametrised conditional distribution $p_\phi(z|x)$ and the prior $p(z)$. The second mutual information term $I_{\phi,\theta}(Z;Y)$ depicts the decoder of our model and is formalised as:

$$I_{\phi,\theta}(T;Y) = \mathbb{E}_{p(x,y)} \mathbb{E}_{p_\phi(z|x)} \log p_\theta(y|z) + h(Y), \tag{4.2.3}$$

We denote with $h(y) = -\mathbb{E}_{p(y)}[\log p(y)]$ the *entropy* for discrete y and the *differential entropy* for continuous y.

We assume a conditional independence copula and Gaussian margins for the conditional distributions which have been defined in the previous Equations 4.2.2 and 4.2.3: The conditional distribution $p_\phi(z|x)$ of the encoder is written as:

$$\begin{aligned} p_\phi(z|x) &= c_{Z|X}(u(z|x)|x) \prod_j p_{\phi_j}(z_j|x) \\ &= \prod_j N(z_j|\mu_j(x), \sigma_j^2(x)), \end{aligned}$$

where z_j is the jth marginal distribution of $z = (z_1, \dots, z_d)$, $c_{z|x}$ is the copula density of $z|x$. In addition, $u(z|x) := F_{z|x}(z|x)$ represents the uniform density indexed by $z|x$ and $\mu_j(x), \sigma_j^2(x)$ denote non-linear functions which are implemented by neural networks.

We also assume an independence copula with Gaussian marginals for the conditional distribution $p_\theta(y|t)$ of the decoder which is formulated as:

$$\begin{aligned} p_{\phi,\theta}(y|z) &= c_{Y|Z}(u(y|z)|z) \prod_j p_{\phi_j,\theta_j}(y_j|z) \\ &= \prod_j N(y_j|\mu_j(z), \sigma_j^2(z)), \end{aligned}$$

An important property of mutual information is the invariance against strictly monotone transformations. In the subsequent section, we analyse the problems if a model does not encode invariances to strictly monotone transformations and the resulting implications for mutual information.

4.2.1 Violating the Invariance Property

As we stated in Section 4.1, the mutual information estimates in the deep information bottleneck model derived in Section 4.2 are not invariant to strictly increasing transformations. In the context of this thesis, the mutual information denotes the symmetry whereas the strictly increasing transformations represent the symmetry transformation g. However, the function f, which calculates the mutual information, is known but can be only observed through pointwise samples. Naively learning a mutual information with a neural network results in severe problems on which we elaborate below. The key issue is that mutual information $I(x, y)$ depends only on the copula and therefore does not depend on monotone transformations of the marginals: $I(x, y) = MI(x, y) - MI(x) - MI(y)$, where $MI(x)$, for $x = (x_1, \dots, x_d)$, denotes the multi-information, which is equal to the negative copula entropy, as shown by Ma & Sun (2011):

$$\begin{aligned} MI(X) &:= D_{KL}(p(x) \| \prod_j p_j(x_j)) \\ &= \int c_X(u(x)) \log c_X(u(x)) \mathrm{d}u \\ &= -h(c_X(u(x))). \end{aligned} \tag{4.2.4}$$

In the following, we will discuss the problems which occur if we learn a function f that is not invariant to strictly increasing transformations in the context of a deep information bottleneck:

Problem 1: Encoder is not flexible enough

On the encoder side (Eq. (3.5.2)), the optimisation is performed over the parametric conditional margins $p_\phi(t_j|x)$ in $I_\phi(t; x) = \mathbb{E}_{p(x)} D_{KL}\big(p_\phi(t|x) \| p(t)\big)$. When a monotone transformation $x_j \to \tilde{x}_j$ is applied, the required invariance property can only be guaranteed if the model for ϕ (in our case a deep network) is flexible enough to compensate for this transformation, which can be a severe problem in practice as it results in wrong mutual information estimates.

Problem 2: Gaussian margins might be inappropriate

On the decoder side, assuming Gaussian margins in $p_\theta(y_j|t)$ might be inappropriate for modelling y if the domain of y is not equal to the real numbers, e.g. when y is defined only on a

bounded interval. If used in a generative way, the model might produce samples outside the domain of y. Even if other distributions than Gaussian are considered, such as truncated Gaussian, one still needs to make assumptions concerning the marginals.

Problem 3: Entropy is not constant for continuous domains

Also on the decoder side, we have defined the mutual information as log-likelihood and an entropy term: $I_\phi(t;y) = \mathbb{E}_{p(x,y)}\mathbb{E}_{p_\phi(t|x)} \log p_\theta(y|t) + h(y)$. The authors of the Deep Information Bottleneck (Alemi et al., 2017) argue that since $h(y)$ is constant, it can be ignored in computing $I_\phi(t;y)$. This is true for a fixed or for a discrete y, but not for the class of monotone transformations of y. Since the left hand side of this equation ($I_\phi(t;y)$) is invariant against monotone transformations, and $h(y)$ in general depends on monotone transformations, the first term on the right hand side ($\mathbb{E}_{p(x,y)}\mathbb{E}_{p_\phi(t|x)} \log p_\theta(y|t)$) cannot be invariant to monotone transformations. In fact, under such transformations, the differential entropy $h(y)$ can take any value from $-\infty$ to $+\infty$, which can be seen easily by decomposing the entropy into the copula entropy and the sum of marginal entropies (here, j stands for the jth dimension):

$$\begin{aligned} h(y) &= h(c_y(u(y))) + \sum_j h(y_j) \\ &= -MI(y) + \sum_j h(y_j). \end{aligned} \tag{4.2.5}$$

The first term (i.e. the copula entropy which is equal to the negative multi-information, as in Eq. (4.2.4)) is a non-positive number. The marginal entropies $h(y_j)$ can take any value when using strictly increasing transformations (for instance, the marginal entropy of a uniform distribution on $[a,b]$ is $\log(b-a)$). As a consequence, the entropy term $h(y)$ in Eq. (3.5.2) can be treated as a constant only for one specific y or for discrete y, but not for all elements of the equivalence class containing all monotone transformations of y. Moreover, every such transformation would lead to different $(I(x,t), I(y,t))$ pairs in the information curve, which basically makes this curve arbitrary. Thus, $h(y)$ being constant is a property that needs to be restored.

4.2.2 Deep Copula Information Bottleneck

In this part, we introduce an approach to learn the function f that is invariant against any monotone increasing transformation g and overcomes the problems described in Section 4.2.1. To this end, we apply a variable transformation for random variables $X = (X_1, \ldots, X_d)$ where d denotes the number of dimension and X_j stands for the jth dimension. Moreover, Φ is the Gaussian cdf and $\hat{F}$ is the empirical cdf. The variable transformation is performed as follows: In the first step, we apply the inverse Gaussian cdf to the empirical cdf in order to obtain a random variable $\tilde{X}$ with Gaussian marginals:

$$\tilde{x}_j = \Phi^{-1}(\hat{F}(x_j)), \tag{4.2.6}$$

In the second step, we apply the inverse empirical cdf to map $\tilde{X}$ back to the initial random variable X.

$$x_j = \hat{F}^{-1}(\Phi(\tilde{x}_j)), \tag{4.2.7}$$

In the copula literature, these transformed random variables are also knows as *normal scores*. It is important to note that the mapping is (approximately) invertible: $x_j = \hat{F}^{-1}(\Phi(\tilde{x}_j))$, with $\hat{F}^{-1}$ being the empirical quantiles treated as a function. This may be achieved by linear interpolation.

However, the distribution F is known in most cases. In order to overcome this limitation, we employ the empirical marginal distribution which can be obtained as follows (Rey & Roth, 2012):

$$\hat{F}(x) = \frac{rank(x)}{n+1} \tag{4.2.8}$$

That is, Eq: 4.2.8 defines a step function between the values $1/n+1$ and $n/n+1$ by evaluating the ranks of the n samples. Note that using a rank transformation has several advantages compared to competing approaches as it is for example robost against outliers. This stems from the fact that extreme values can only lie within the range 1 to n why the ouliers cannot skew results that much.

Example: Calculating Normal Scores

We give an intuition how the calculation of our method works in practice. To this end, we transform an empirical distribution to a Gaussian distribution based on the probability integral transform (see Section 3.3.2). Consider the following samples drawn from an unknown distribution F_X:

$$x = \{62.1, 0.25, 0.41.0.02, 1.36, 0.76, 1.63\}$$

To perform the copula transformation, we have to calculate the ranks of x:

$$rank(x) = \{7, 2, 3, 1, 5, 4, 6\}$$

and subsequently obtain the empirical distribution $\hat{F}$ (Eq: 4.2.8):

$$\hat{F}(x) = \{0.875, 0.25, 0.375, 0.125, 0.625, 0.5, 0.75\}$$

In the last step, we transform the empirical distribution to Gaussian with the inverse Gaussian cdf:

$$\tilde{x} = \Phi^{-1}(\hat{F}(x_j)) = \{1.15, -0.67, -0.31, -1.15, 0.31, 0.00, 0.67\}$$

The proposed solution fixes the invariance problem on the encoding side that was described in Problem 1 in Section 4.2.1. In addition, it also resolves the issue on the decoding side (Problem 2) because the transformed variables $\tilde{x}_j$ are standard normal distributed. In addition, our solutions also resolves Problem 3 because the decoder part (Eq. (4.2.3)) admits the following form:

$$\begin{aligned} \mathbb{E}_{p(\tilde{x},\tilde{y})}\mathbb{E}_{p_\phi(t|\tilde{x})} \log p_\theta(\tilde{y}|t) &= I_\phi(t;\tilde{y}) + MI(\tilde{y}) - \sum_j h(\tilde{y}_j) \\ &= I_\phi(t;\tilde{y}) - h(c_{\text{inv}}(u(\tilde{y}))) \end{aligned} \tag{4.2.9}$$

where $c_{\text{inv}}(u(\tilde{y}))$ is indeed constant for all strictly increasing transformations applied to y.

After we have learned the function f to calculate the mutual information in the transformed space, we can go back to the original space by using the inverse transformation according to Equation (4.2.7) $x_j = \hat{F}^{-1}(\Phi(\tilde{x}_j))$.

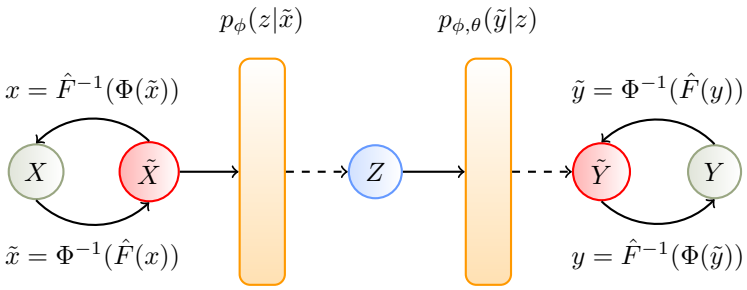


Figure 4.3: Deep information bottleneck with the copula augmentation. Green circles describe random variables and orange rectangles denote neural networks parametrising the random variables. The blue circle represents latent random variables whereas the red circle denotes the copula transformed random variables.

4.2.3 Implementation and Training Procedure

In this section, we describe the implementation of the mutual information terms in practise. To this end, the encoder part $I_\phi(Z;\tilde{X})$ where ϕ denote the learnable parameters is calculated as follows: We compute the mutual information term as KL divergence $D_{KL}\left(p_\phi(z|\tilde{x})\|p(z)\right)$ between $p_\phi(z|\tilde{x})$ and a simple Gaussian prior $p(t) = \mathcal{N}(t;0,I)$. For simplicity, we represent $p_\phi(z|\tilde{x})$ as Gaussian distribution with parameters μ and σ that are learned with a neural network that is parametrised by ϕ. As we have defined both parts of the KL-divergence as Gaussian distributions, the KL-divergence admits an analytical form.

$$\begin{aligned} I_\phi(Z;\tilde{X}) = &\ \mathbb{E}_{p(\tilde{x})} D_{KL}\left(p_\phi(z|\tilde{x})\|p(z)\right) \\ \approx &\ \frac{1}{n}\sum_i D_{KL}\left(p_\phi(z|\tilde{x}_i)\|p(z)\right) \end{aligned} \tag{4.2.10}$$

Subsequently, we make use of the reparametrisation trick introduced by Kingma & Welling (2014). In order to sample from the Gaussian distribution, we take the learned parameters of the μ and σ and and reformulate it in the following way: $\mu + \sigma \cdot \epsilon$ where ϵ is drawn from $\mathcal{N}(0,1)$.

For the decoder side, $\mathbb{E}_{p(\tilde{x},\tilde{y})}\mathbb{E}_{p_\phi(z|\tilde{x})}\log p_\theta(\tilde{y}|z)$ is implemented as follows:

$$\begin{aligned} I_{\phi,\theta}(Z;\ \tilde{Y}) = &\ \mathbb{E}_{p(\tilde{x},\tilde{y})}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)} \\ &\sum_j \log p_\theta(\tilde{y}_j|t = \vec{\mu}_j(\tilde{x}) + diag(\sigma_j(\tilde{x}))\cdot\epsilon) + \text{const.}, \end{aligned} \tag{4.2.11}$$

Here, j denotes the j^{th} data sample and the conditional distribution $p_{\phi,\theta}(y|z)$ is implemented as a neural network with parameters θ.

Subsequently, we illustrate the training procedure for our model. A detailed description is given in Algorithm 1. We employ the previously described copula transformations of the input X and Y in lines 1 and 2. After, we train our model by using stochstic gradient descent. For every epoch, we sample i minibatches, where the number of minibatches i is determined by the batchsize. In the first step of the training algorithm, we estimate the mutual information between $\tilde{X}$ and Z by encoding the input $\tilde{X}$ in the latent space Z. Subsequently, we decode the latent Z to the output $\tilde{Y}$ in line 8. After, we update the parameters ϕ and θ by taking a gradient step. We increase the compression parameter λ by a predefined factor l after every epoch.

Algorithm 1 Deep Copula Information Bottleneck

Input: input X, target Y
$\tilde{x} = \Phi^{-1}(\hat{F}(x))$
$\tilde{y} = \Phi^{-1}(\hat{F}(y))$
for each epoch **do**
  sample i minibatches of $\tilde{x}$ and $\tilde{y}$
  for each minibatch i **do**

    encode $\tilde{x}_i$ into $p_\phi(z_i \mid \tilde{x}_i)$
    decode z_i to obtain $p_\theta(\tilde{y}_i \mid z_i)$

    update ϕ, θ by taking a gradient step

  end for
  increase λ by factor l
end for

4.3 Experiments

We now proceed to experimentally verify the effect of the Deep Copula Information Bottleneck to test the impact of the copula transformation. To this end, we perform a series of pair-wise experiments, where the Deep Information Bottleneck is tested with and without the copula transformation. We use both an artificial and real-world and devise multiple experimental setups.

4.3.1 Artificial Dataset

We construct an artificial dataset of samples X and Y and try to compress X in a latent variable Z which should only retain information about the output variable Y. In order to achieve the reconstruction of Y, Z requires to be a high-dimensional latent space. In order to verify our approach, we perform monotone transformations on this dataset and test the difference between Deep Information Bottleneck and the Deep Copula Information Bottleneck on reconstruction capabilities as well as classification predictive score.

Dataset.

The model used to generate the data consists of two input vectors x_1 and x_2 drawn form a uniform distribution defined on $[0,2]$ and vectors k_1 and k_2 drawn uniformly from $[0,1]$. Additional inputs are $x_{i=3\ldots10} = a_i * k_1 + (1-a_i) * k_2 + 0.3 * b_i$ with a_i, b_i drawn from a uniform distribution defined on $[0,1]$. All input vectors $x_{1\ldots10}$ form the input matrix X. Latent variables $z_1 = \sqrt{x_1^2 + x_2^2}$ and $z_2 = z_1 + x_4$ are defined and then normalised by dividing through their maximum value. Finally, random noise is added. Two target variables $y_1 = z_2 * \cos(1.75 * \pi * z_1)$ and $y_2 = z_2 * \sin(1.75 * \pi * z_1)$ are then calculated. y_1 and y_2 form a spiral if plotted in two dimensions. The angle and the radius of the spiral are highly correlated. Therefore, a one-dimensional latent space can only reconstruct the backbone of the spiral. In order to reconstruct the details of the radial function, one has to use a latent space of at least two dimensions. We generate 200k samples from X and Y. X is further transformed to beta densities using strictly increasing transformations. We split the samples into test (20k samples) and training (180k samples) sets. The generated samples are then transformed with the copula transformation (Eq. (4.2.6)) to $\tilde{X}$ and $\tilde{y}$ and split in the same way into test and

training sets. This gives us the four input sets X_{train}, X_{test}, $\tilde{X}_{train}$, $\tilde{X}_{test}$ and the four target sets y_{train}, y_{test}, $\tilde{y}_{train}$, $\tilde{y}_{test}$.

Setup.

We define the input and the output layer to have 10 and 2 dimensions, respectively. In addition, specify a latent layer with ten nodes that model the means of the ten-dimensional latent space Z. In contrast to other implementations where a variance is also learned by the model, we set the variance of the latent representation Z to be 1 for simplicity. The encoder part of our model (I) as well as the decoder part ($I_{\phi,\theta}$) are implemented as a neural network with neural network parameters ϕ and θ. Specifically, we employ with two fully-connected hidden layers with 50 nodes for each neural network. In addition, we use a softplus function as the activation function. In order to train our model, we use stochastic gradient descent (Adam optimiser (Kingma & Ba, 2015)) together with a batch size of 500. We train the model for 70000 iterations using a learning rate of 0.0006. We increase the compression parameter λ by multiplying λ (Equation (4.2.1) with 1.06 every 500 iterations to obtain an information curve.

Experiment 1: Information Curves

We compare the information curves produced by the Deep Information Bottleneck and its copula augmentation (Figure 4.4a). To this end, we use the training sets (X_{train}, Y_{train}) and ($\tilde{X}_{train}$, $\tilde{Y}_{train}$) to train our models. In order to obtain the information curves, we record the values of $I(X;Z)$ and $I(Z;Y)$ during the optimisation process. To do so, we multiply the λ parameter every 500 iterations by 1.06 during training. Afterwards, we bin the recorded mutual information estimates into 12 different bins for better visualisation. In our experiment, we can observe an increase in the mutual information of $I(Z;Y)$ from approximately 6 in the Deep Information Bottleneck to approximately 11 after we applied the copula transformation to the data. At the same time, the mutual information of $I(X;Z)$ raise from 6.6 to 16.3 for the Deep Information Bottleneck. This clearly indicates the an mutual information estimator based on plain neural networks cannot account for monotone transformations as the mutual information estimates are lower compared to the estimates after the copula transformation. Thus, such an estimator leads to wrong mutual information estimates that does not preserve the invariance properties of mutual information under such symmetry transformations. However, by applying a copula transformation to the marginal distribution, we can guarantee that the mutual information only depends on the copula which leads to correct mutual information estimates that preserve the required invariance properties.

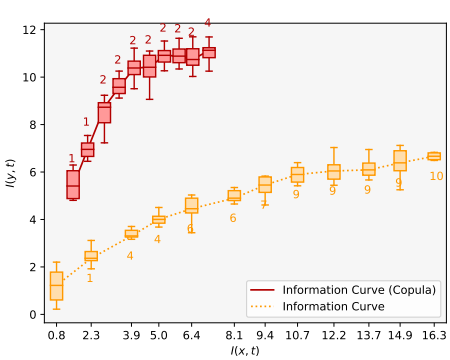
Experiment 2. Evaluating the Predictive Mutual Information

Building on Experiment 1, we investigate how wrong mutual information estimates affect the predictive quality of the trained models. To do so, we evaluate the Deep Information Bottleneck and its copula counterpart on test data (X_{test}, y_{test}) and ($\tilde{X}_{test}$, $\tilde{y}_{test}$) where the latter test dataset is copula transformed. We compute predictive scores of the latent space Z with respect to the generated Y in the form of mutual information $I(Z;Y)$ for all values of the parameter λ. The resulting information curve shows an increased predictive capability of the Deep Copula Information Bottleneck in Figure 4.4b and exhibits no difference to the information curve produced in Experiment 1 (Figure 4.4a). Thus, the increased mutual information reported in Experiment 1 cannot only be attributed to overfitting and therefore supports our finding the plain neural

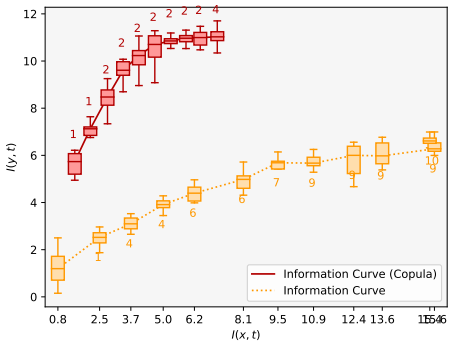
networks cannot estimate mutual information which is invariant under monotone increasing transformations.

Experiment 3. Reconstruction Capability

We qualitatively assess the reconstruction capability of the Deep Copula Information Bottleneck compared to its plain counterpart in Figure 4.5. We choose the value of λ such that in both models are using two dimensions in the latent space to reconstruct the output Y. In the case where we apply the copula transformation to the data, our model is capable of learning a detailed reconstruction of the output Y which is depicted in Figure 4.5b. In contrast, the reconstruction quality of the Deep Information Bottleneck on test data results merely in a tight backbone which is not capable of reconstructing Y (Figure 4.5a). This supports our findings in Experiment 1, that a plain neural network cannot learn mutual information estimates that are invariant against strictly increasing transformations and thus lead to wrong mutual information estimates.



(a) Training Information Curve

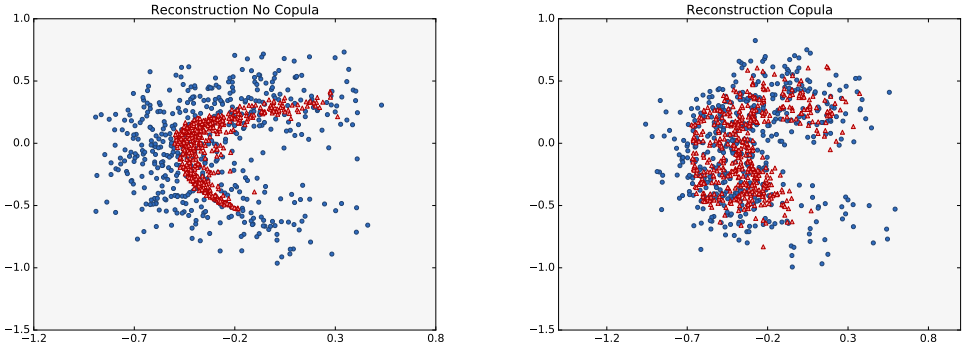


(b) Predictive Information Curve (evaluated on test data)

Figure 4.4: Information curves for the artificial experiment. The red curve describes the information curve with copula transformation whereas the orange one illustrates the plain information curve. For better visualisation, we binned the recorded mutual information values into 12 different buckets. The numbers indicate the used latent dimensions.

Experiment 4. Robustness against Outliers and Adversarial Attacks

We further inspect the information curves of the Deep Information Bottleneck and the Deep Copula Information Bottleneck by testing how the copula transformation adds resilience of the model against outliers and adversarial attacks in the training phase. To this end, we simulate an adversarial attack with the following procedure: We randomly choose 5% of all entries in the input X_{train} and replace them with outliers by adding uniformly sampled noise within the range [1,5]. Subsequently, we perform a copula transformation on the modified dataset X_{train} to obtain the second dataset $\tilde{X}_{train}$. Now, we again compute information curves for the training procedure and compare normal training with training with data subject to an attack for the copula and non-copula models. Here, we receive comparable information curves to Experiment 1 (Figure 4.4a). In addition, the results (Figure 4.6) showcase that the copula model is more robust against outlier data than the plain one. We attribute this behaviour directly to the copula transformation, as



(a) (b)

Figure 4.5: illustrates the reconstruction of Y without (a) and with the copula transformation (b). Blue circles depict the output space and the red triangles — the conditionals means $\mu(y)$ that are estimated by our network. The better the red triangles reconstruct the blue area, the better is the reconstruction of the output.

ranks are less sensitive to outliers than raw data.

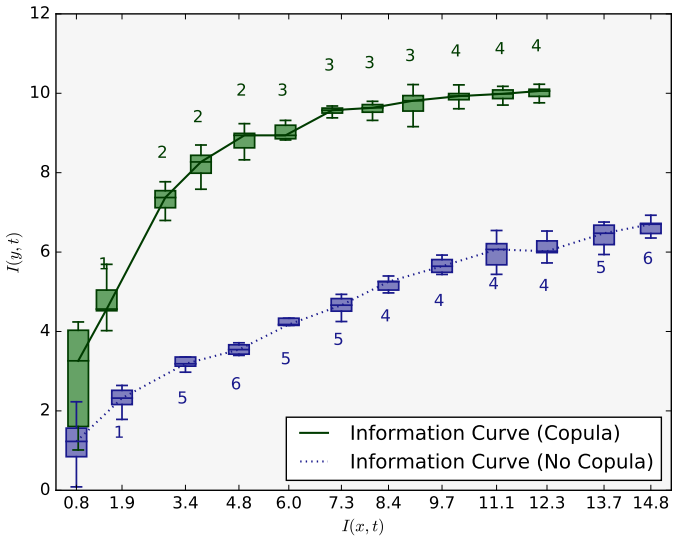


Figure 4.6: Information curves for training with outlier data and a sample convergence plot of DIB and cDIB models for $\lambda = 100$. The numbers indicate the used latent dimensions.

4.3.2 Communities and Crime Dataset

We continue analysing the impact of the copula transformation to estimate mutual information which are invariant to strictly monotone transformations in a real-world setting. We first report information curves analogous to Experiment 1 (Section 4.3.1) and proceed to inspect the latent spaces of both models along with sensitivity analysis with respect to λ.

Dataset.

In this experiment, we consider the unnormalised *Communities and Crime* dataset from the UCI repository[1]. This dataset deals with crimes crime rates within communities in the United States of America. To do so, the dataset integrates covariates from three different data sources: namely, socio-economic data from the 1990 US Census, law enforcement data from the 1990 US LEMAS survey, and crime data from the 1995 FBI UCR. In total, the dataset consists of 125 predictive, 4 non-predictive and 18 target variables with 2215 samples in total. In a preprocessing step, we removed all missing values from the dataset. In the end, we used 1901 observations with 102 predictive and 18 target variables in our analysis.

Setup.

The set-up of this experiment is defined as follows: We employ a 102 dimensional input and a 18 dimensional out for our model. In addition, we use a latent layer with 18 nodes that models the means of the 18-dimensional latent space Z. For simplicity, we do not learn the variance of Z but set to variance to be 1 in advance. The encoder term I_ϕ as well as the decoder term $I_{\phi,\theta}$ consist of a neural network with parameters ϕ and θ. Both neural networks are defined as two fully-connected hidden layers with 100 nodes each. We use a Softplus function as the nonlinear activation function for the neural network. The decoder uses a Gaussian likelihood where we set the variance to 1 which thus results in a least-squares loss. After having defined our model, we use stochastic gradient descent (Adam (Kingma & Ba, 2015)) with a learning rate of 0.0005 and a batch size of 1255 to train our model. In total, we train the model for 150000 iterations and increase the compression parameter λ sequentially. To do so, we multiply λ by 1.01 every 500 iterations.

Experiment 6. Information Curves

This experiment is analogous to Experiment 1 (Section 4.3.1) where we calculate information curves from the Deep Information Bottleneck and the Deep Copula Information Bottleneck. To do so, we again record the mutual information estimates of $I(x;t)$ and $I(y;t)$ while multiplying the λ parameter every 500 iterations by 1.01 during training. Here, the information curve for the copula model yields larger values of the mutual information $I(Z,Y$ and lower values of $I(X,Z$. For the mutual information $I(Z,Y$, we obtain an increase of approximately 0.5 whereas $I(X,Z$ shows a decrease of approximately 4. That is, we obtain more consistent mutual information estimates if we apply a copula transformation. We attribute this effect to the increased flexibility of the model, as we pointed out in Section 4.2.2. In addition, to our qualitative comparison, we perform quantitative comparisons to demonstrate that our information curves are significantly different. To do so, we employ a statistical Kruskal-Wallis rank test. A Kruskal-Wallis test performs a non-parametric test if two samples are not drawn from the same distribution. To compare two distributions, we select two bins in our plot which lie on approximately the same point on the x-axis. Therefore, we select the fifth bin of the copula and the fourth bin the non-copula curve. When we compare the previously described bins with the Kruskal-Wallis test, we obtain a significant difference with a p-value of $1.6 * 10^{-16}$.

[1] http://archive.ics.uci.edu/ml/datasets/communities+and+crime+unnormalized

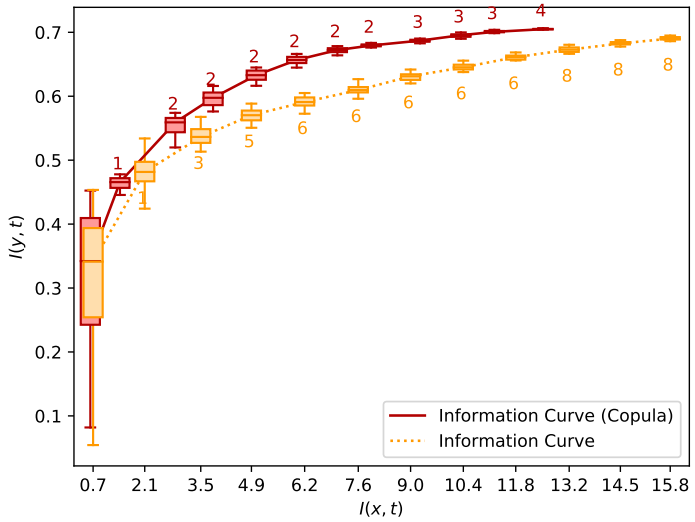


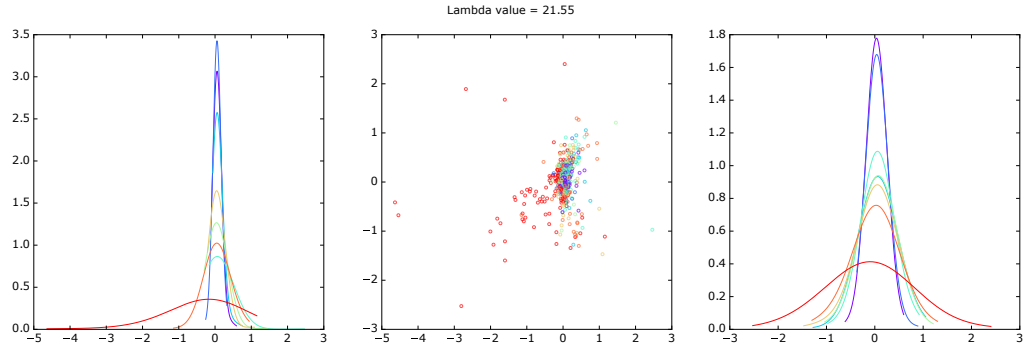
Figure 4.7: Information curves for the real data experiment. The red curve is the information curve with copula transformation whereas the orange one depicts the plain information curve. The numbers represent the dimensions in the latent space t which are needed to reconstruct the output y. The numbers indicate the used latent dimensions.

Experiment 7. Structured Representations

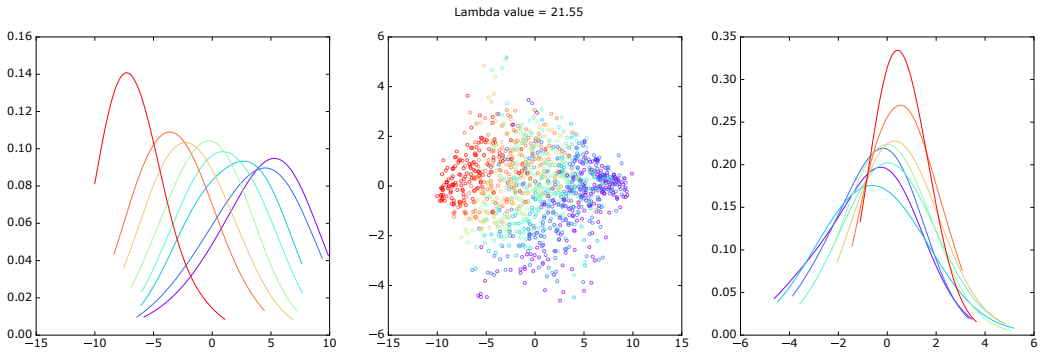
The last experiment illustrates the difference between the learned representations of the latent spaces of the Deep Information Bottleneck models with and without the copula transformation. We select two variables which yield to highest correlation with the target variable *arsons* and plot them along with their densities. In order to obtain the corresponding class labels (rainbow colours in Figure 4.8), we separate the values of *arsons* in eight equally-sized bins. A sample comparison of latent spaces of Deep Information Bottleneck and the Deep Copula Information Bottleneck for $\lambda = 21.55$ is depicted in Figure 4.8. The latent space Z of Deep Information Bottleneck (middle figure in Figure 4.8a) appears consistently less structured than the latent space of Deep Copula Information Bottleneck in the middle figure of Figure 4.8b. This is, the mutual information estimates without copula are significantly worse and can only be solved by applying a copula transformation. This observation is also reflected in the 1D density plots of the variables which are depicted in the left and right image of Figure 4.8a and 4.8b. In Figure 4.8b with the copula transformation, we can identify a much clearer structure in the latent space with respect to our previously calculated class labels which supports the assumption that the Deep Information Bottleneck leads to erroneous mutual information estimates.

4.4 Summary

In this chapter, we addressed the problem incorporating invariances in deep generative models when the corresponding symmetry transformation g is known. Specifically, we focused on the information bottleneck principle where the invariant feature extractor f is the mutual information and the symmetry transformation is the class of all monotone increasing transformation. Therefore in theory, the mutual information estimates should be constant for all monotone transformations such as a transformation of the marginal distribution from Gaussian to Beta.



(a)



(b)

Figure 4.8: We illustrate the latent space Z which consists of two dimensions along with marginal densities without (a) and with (b) the copula transformation. The copula transformation leads to better mutual information estimates as we obtain a more structured latent space and non-overlapping modes of marginal distributions.

In the past, there have been advances which solved this problem for the linear version of the information bottleneck principle by introducing a Gaussian copula with arbitrary margins (Rey & Roth, 2012). More recently, Alemi et al. (2017) introduced a non-linear extension of the information bottleneck where the calculation of the function f is approximated by a neural network. However in this approach, the mutual information is not invariant under the class of monotone increasing transformations which leads to erroneous mutual information estimates.

In this chapter, we performed a theoretical analysis of Deep Information Bottleneck and demonstrated that neural networks are not invariant against monotone increasing transformations. To this end, we introduced a novel method that overcomes the described limitations by introducing a copula transformation. In particular, we transform an arbitrary marginal distribution with the concept of normal scores to a Gaussian distribution. Subsequently, we are able calculate the mutual information as the Deep Information Bottleneck is defined in the Gaussian setting. As a result, the proposed model allows for a simplified and fully non-parametric treatment of marginal distributions which has the advantage that it can be applied to distributions with arbitrary marginals.

Our experiments confirm, that the Deep Information Bottleneck suffers from the lack of invariance against monotone transformation, thus leading to inconsistent mutual information estimates. In particular, we demonstrated that variant mutual information estimates result in lower mutual information curves and unstructured latent representations, amongst others. In contrast,

estimating mutual information with the copula transform leads to reliable mutual information estimates and structured latent representations on both artificial and real-world datasets.

Chapter 5

Learning Symmetry Transformations using Mutual Information Regularisation

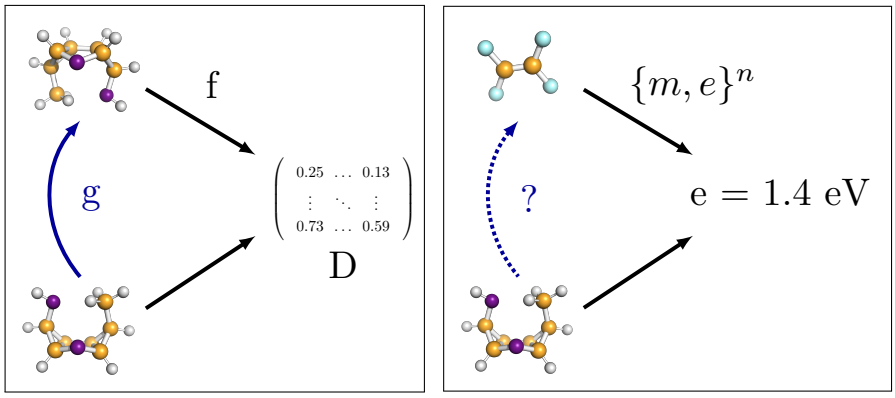
This chapter has already been accepted at the Neural Information Processing Systems (NeurIPS) conference. Thus, this work closely follows Wieser et al. (2020).

5.1 Introduction

In this chapter, we aim to solve the second research question of this thesis where the symmetry transformation g is unknown and therefore needs to be learned from data. Again, reconsider the example of rotational invariance from Figure 5.1a. Here, we show the 3D representation of a molecule m which is rotated using a function g. Subsequently, we calculate the distance matrix D between the atoms of the rotated molecule $g(m)$ using a predefined function f. As a result, we obtain the same distance matrix for every rotation g, i.e. $f(m) = f(g(m))$ for any rotation g.

However, it is challenging or impossible to find analytical forms of symmetry transformations g in highly complex domains such as the chemical space (Figure 5.1b). As a running example, we reconsider the task of discovering novel molecules for the design of organic solar cells in material science. In this case, all molecules must possess specific properties, e.g. a bandgap energy of approximately 1.4 eV (Shockley & Queisser, 1961). In such scenarios, there is no predefined symmetry transformation (such as rotation) known or can be assumed. Yet, we can observe available data defining our invariance class from $\{m, e\}^n$ numeric point-wise samples from the function f where n is the number of samples, m the molecule and $e = f(m)$ the bandgap energy.

The goal of our model is thus to learn the class of symmetry transformations g which result in a symmetry property f of the modelled system. To this end, we learn a continuous data representation and the corresponding symmetry transformation in an inverse fashion from data samples $\{m, e\}_1^n$ only. To do so, we introduce the Symmetry-Transformation Information Bottleneck (STIB) where we encode the input X (e.g. a molecule) into a latent space Z and subsequently decode it to X and a preselected target property Y (e.g. the bandgap energy). Specifically, we divide the latent space into two subspaces Z_0 and Z_1 to explore the variations of the data with respect to a specific target. Here, Z_1 is the subspace that contains information about input and target, while Z_0 is the subspace that is invariant to the target. In doing so, we capture symmetry transformations not



(a) Rotational symmetry transformation.

(b) Unknown symmetry transformation.

Figure 5.1: In Figure 5.1a a molecule is rotated by g. The distance matrix D between atoms is calculated by a known function f and remains unchanged for all rotations. In Figure 5.1b we can only observe point-wise samples n samples $\{m,e\}^n$ where m is the molecule and e the bandgap energy. These samples approximate the function f whereas the class of functions g leading to the same bandgap energy is unknown.

affecting the target Y in the isolated latent space Z_0.

The central element of STIB is minimising the information about continuous Y (e.g. bandgap energy) present in Z_0 by employing adversarial learning. In contrast, cognate models have to the best of our knowledge solely focused on discrete Y. The potential reason is that naively using the negative log-likelihood (NLL) as done for maximising mutual information in other deep information bottleneck models leads to critical problems in continuous domains. This stems from the fact that fundamental properties of mutual information, such as invariance to one-to-one transformations, are not captured by this mutual information estimator. Simple alternatives such as employing a coarse-grained discretisation approach as proposed in Robert et al. (2019) are not feasible in our complex domain. The main reason is that we want to consider multiple properties at once, every one of which might require a high-resolution. Simultaneous high-resolutional discretisation of multiple targets would result in an intractable classification problem.

To overcome the aforementioned issues, we propose a new loss function based on Gaussian mutual information with a bijective variable transformation as an addition to our modelling approach. In contrast to using the MSE, this enables the calculation of the mutual information on the basis of correlations. With this, we ensure a well defined loss function and invariance against linear one-to-one transformations.

In summary, we make the following contributions:

1. We introduce a deep information bottleneck model that learns a continuous low-dimensional representation of the input data. We augment it with an adversarial training mechanism and a partitioned latent space to learn symmetry transformations based on this representation.

2. We further propose a continuous mutual information regulation approach based on correlation matrices. This makes it possible to address the issue of one-to-one transformations in the continuous domain.

3. Experiments on an artificial as well as two molecular datasets demonstrate that the pro-

posed model learns both pre-defined and arbitrary symmetry transformations and outperforms state-of-the-art methods.

5.2 Adversarial Information Elimination

A common approach to remove information from latent representations in the context of VAEs is using adversarial training (Creswell et al., 2017; Klys et al., 2018; Lample et al., 2017). The main idea is to train an auxiliary network $a_\psi(z)$ which tries to correctly predict an output b from the latent representation z by minimising the classification error. Concurrently, an adversary, in our case the encoder network of the VAE, tries to prevent this. To this end, the encoder $q_\theta(z|x)$ attempts to generate adversarial representations z which contain no information about b by maximising the loss $\mathscr{L}$ with respect to parameters θ. The overall problem may then be expressed as an adversarial game where we compute:

$$\max_\theta \min_\psi \mathscr{L}(a_\psi(q_\theta(z|x)), b), \tag{5.2.1}$$

with $\mathscr{L}$ denoting the cross-entropy loss. While this approach is applicable for discrete domains, generalising this loss function to continuous settings can lead to severe problems in practice. We elaborate on this issue in the next section.

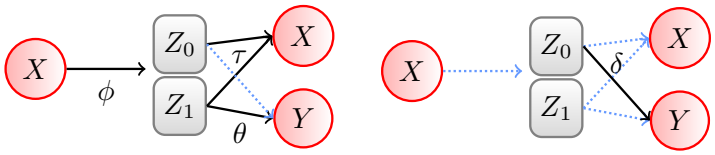


Figure 5.2: Graphical illustration of our two-step adversarial training approach. Red circles denote observed input/output of our model. Gray rectangles represent the latent representation which is divided into two separate subspaces Z_0 and Z_1. Blue dotted arrows represent neural networks with fixed parameters Black arrows describe neural networks with trainable parameters. Greek letters define neural network parameters. In the first step (Figure 6.3a), we try to learn a representation of Z_0 which minimises the mutual information between Z_0 and Y by updating ϕ, θ and τ. In the adversary step (Figure 6.3b), we maximise the mutual information between Z_0 and Y by updating δ.

5.3 Symmetry-Transformation Information Bottleneck

As previously described in Section 5.1, our goal is to learn symmetry transformations g based on observations X and Y (see Figure 5.1b). Here, X and Y may be complex objects, such as molecules and its corresponding bandgap energies, which are difficult to manipulate consistently. In order to overcome this issue, we aim to learn a continuous low-dimensional representation of our input data X and Y in Euclidian space. To do so, we augment the traditional deep information bottleneck formulation (Eq. (3.5.1)) with an additional decoder reconstructing X from Z. Our base model is thus defined as an augmented parametric formulation of the

information bottleneck (see Eq. (3.5.1)):

$$\min_{\phi,\theta,\tau} I_\phi(Z;X) - \lambda(I_{\phi,\theta}(Z;Y) + I_{\phi,\tau}(Z;X)), \tag{5.3.1}$$

where ϕ,θ,τ describe neural network parameters and λ the compression factor.

5.3.1 Theoretical Concept.

Based on the model specified in Eq. 5.3.1, we provide a novel approach to learn symmetry transformations on latent representations Z. To this end, we propose partitioning the latent space Z into two components, Z_0 and Z_1. Z_1 is intended to capture all information about Y, while Z_0 should contain all remaining information about X. That is, changing Z_0 should change X but *not affect* the value of Y. Thus, Z_0 expresses a surrogate of the unknown function g which is depicted in Figure 5.1b. However, simply partitioning the latent space is not sufficient, since information about Y may still be encoded in Z_0.

To address this task, we propose combining the model defined in Eq. 5.3.1 with an adversarial approach (Section 5.2). The resulting model thus reduces to playing an adversarial game of minimising and maximising the mutual information $I_\delta(Z_0,Y)$ where δ denotes the neural network parameters. This ensures that Z_0 contains no information about Y. In more detail, our approach is formulated as follows:

In the first step (see Figure 5.2a), our model learns a low-dimensional representation Z_0 and Z_1 of X and Y by maximising the mutual information between $I_{\phi,\tau}(Z_0,Z_1;X)$ and $I_{\phi,\theta}(Z_1;Y)$. At the same time, our algorithm tries to eliminate information about Y from Z_0 by minimising the mutual $I_\delta(Z_0;Y)$ via changing Z_0 with fixed parameters δ (brown part of Eq. 5.3.2).

$$\mathscr{L}_1 = \min_{\phi,\theta,\tau} I_\phi(X;Z) - \lambda\Big(I_{\phi,\tau}(Z_0,Z_1;X) + I_{\phi,\theta}(Z_1;Y) - I_\delta(Z_0;Y)\Big) \tag{5.3.2}$$

The second step defines the adversary of our model and is illustrated in Figure 5.2b. Here, we try to maximise the mutual information $I_\delta(Z_0;Y)$ (purple part of Eq. 5.3.3) given the current representation of Z_0. To do so, we fix all model parameters except of δ and update the parameters accordingly.

$$\mathscr{L}_2 = \min_{\delta} I_\phi(X;Z) - \lambda\Big(I_{\phi,\tau}(Z_0,Z_1;X) + I_{\phi,\theta}(Z_1;Y) + I_\delta(Z_0;Y)\Big) \tag{5.3.3}$$

The resulting loss functions $\mathscr{L}_1$ and $\mathscr{L}_2$ are alternately optimised until convergence. Yet, minimising mutual information for continuous variables, such as bandgap energy, remains a challenging task.

Challenges in Continuous Domains.

Mutual information is invariant against the class of one-to-one transformations as it depends only on the copula Ma & Sun (2011). That is, $I(X;Y) = I(f(X);g(Y))$, where g and f denote one-to-one transformations. Related models extending the deep information bottleneck define mutual information for random variables as the NLL plus marginal entropy (see Eq. (3.5.2)). Building on this, only the NLL part of mutual information is optimised. This part alone is, however, not invariant against such transformations. This gives rise to problems in tasks involving minimising mutual information, e.g. as required by our adversary in Eq. (5.3.2). This is because while maximising the NLL, the network can learn solutions stemming from one-to-one transformations of

the marginal. For example, the network might maximise the NLL by adding only a large bias to the output. This leads to an increased NLL even though MI remains unchanged. This results in solutions which are not desired when minimising MI. Therefore, we require a more sophisticated approach that estimates the full mutual information and hence introduces invariance against one-to-one transformations.

Suggested Solution.

We propose a solution that estimates mutual information in a Gaussian setting. This approach is based on correlation matrices and circumvents the problems discussed in the previous paragraph. The reason is that mutual information based on correlation can be meaningfully minimised and is by definition invariant against linear one-to-one transformations. In this setting, mutual information can be decomposed into a sum of multi-informations (Liu, 2012):

$$I(Z_0; Y) = M(Z_0; Y) - M(Z_0) - M(Y), \tag{5.3.4}$$

where the specific multi-information terms are specified as follows:

$$\begin{aligned} M(Z_0, Y) &= \frac{1}{2}\log\left((2\pi e)^{n+m}|R_{Z_0 Y}|\right) \\ M(Z_0) &= \frac{1}{2}\log\left((2\pi e)^{n}|R_{Z_0}|\right) \\ M(Y) &= \frac{1}{2}\log\left((2\pi e)^{m}|R_{Y}|\right), \end{aligned} \tag{5.3.5}$$

where Z_0 and Y are n- and m-dimensional, respectively. $R_{Z_0 Y}$, R_{Z_0} and R_Y denote the sample covariance matrices of $Z_0 Y$, Z_0 and Y, respectively. In practice, we calculate the correlation matrices based on the sample covariance. Note that in the Gaussian setting in which our model is defined, correlation is defined as a deterministic function to the mutual information (Cover & Thomas, 2006).

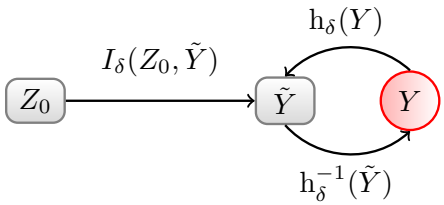


Figure 5.3: Model extended with the bijective mapping between Y and $\tilde{Y}$. Solid arrows depict a nonlinear function parametrised by a neural network. Gray rectangles denote latent and red circles observed variables.

Relaxing the Gaussian Assumption.

As previously stated, to deal with probabilistic models, we require a simple parametric model. Hence, we made the strong parametric assumptions that both Z_0 and Y are Gaussian distributed. However, the target variable Y does not necessarily follow a Gaussian distribution. Our approach to relax this assumption is to open the limited Gaussian distribution to the class of all non-linearly transformed Gaussian distributions. For this reason, we equip the model with a proxy bijective

mapping $Y \leftrightarrow \tilde{Y}$ (Figure 5.3) to introduce more flexibility. This mapping is implemented as two additional networks between Y and a new proxy variable $\tilde{Y}$. The parameters are added to the existing parameters δ. We subsequently treat $\tilde{Y}$ as values to be predicted from Y. We found in our experiments that this approximate approach is sufficient to ensure the invariance property. Our approach makes it possible to compute the mutual information between Z_0 and Y (or its proxy, $\tilde{Y}$) analytically with the formula for Gaussian variables. Thus, we augment $\mathscr{L}_2$ as follows:

$$\mathscr{L}_{\text{bijection}} = \mathscr{L}_2 + \beta \| h_\delta^{-1}(h_\delta(Y)) - Y \|_2^2$$

5.3.2 Implementation

In this section, we describe how to implement the mutual information terms of our model in practice. To do so, we describe the encoder term $I(Z;X)$, which is calculated as the Kullback-Leibler divergence (D_{KL}) between $p_\phi(z|x)$ and $p(z)$. We implement $p_\phi(z|x)$ as a Gaussian distribution with parameters μ and σ that are learned by a neural network with parameters ϕ. Subsequently, we define the second part of the KL divergence $p(z)$ to be a simple Gaussian prior $\mathscr{N}(0,1)$. Due to the fact that we have defined both parts as Gaussian distributions, it is thus possible to calculate the KL divergence analytically.

$$I(Z;X) = \mathbb{E}_{p(x)} D_{KL}(p_\phi(z|x) \| p(z)) \tag{5.3.6}$$

With the first mutual information term (Eq. 5.3.6), we learn a compressed representation Z of the input X. In a next step, we partition Z into two latent spaces Z_0 and Z_1. Here, we assume that Z consists of k latent dimensions where Z_0 contains $h = 0 \dots d$ dimensions of Z where $d < k$. Z_1 thus includes the remaining $m = d+1 \dots k$ dimensions. However upon this point, we have only learned the parameters of the Gaussian distribution. In order to sample from the latent space Z, we make use of the reparametrisation trick introduced in (Kingma & Welling, 2014; Rezende et al., 2014). To do so, a sample from Z can be drawn by reformulating the Gaussian distribution. Therefore, we apply the following formulation: $\mu + \sigma \odot \epsilon$ where ϵ is drawn from $\mathscr{N}(0,1)$ and μ and σ are the learned parameters.

In the last step, we describe how the decoders for X, Y have been implemented by our model. The decoders employ the following form:

$$I(Z_0, Z_1; X) = \mathbb{E}_{p(x)} \mathbb{E}_{\epsilon \sim \mathscr{N}(0,I)} \sum_j \log p_{\tau_j}(x_j | z = \mu_k(x) + \text{diag}(\sigma_k(x)), \odot \epsilon),$$

$$I(Z_1; Y) = \mathbb{E}_{p(x,y)} \mathbb{E}_{\epsilon \sim \mathscr{N}(0,I)} \sum_j \log p_{\theta_j}(y_j | z_1 = \mu_h(x) + \text{diag}(\sigma_h(x)), \odot \epsilon),$$

$$I(Z_0;Y) = \mathbb{E}_{p(x,y)}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I)} \sum_j \log p_\delta(y_j|z_0 = \mu_d(x) + \mathrm{diag}(\sigma_d(x)) \odot \epsilon),$$

where j denotes the j$^{\text{th}}$ data sample. The distributions $p_\tau(x|z)$, $p_\theta(y|z_1)$ and $p_\delta(y|z_0)$ are implemented as neural networks with parameters ϕ, τ, δ. In contrast to the encoder, these distributions may be arbitrarily chosen as we do not make any assumption about their form.

Algorithm 2 Symmetry-Transformation Information Bottleneck

Input: input x, target y

1: **for** each epoch **do**
2: sample i minibatches of x and y
3: **for** each minibatch i **do**
4:
5: Minimisation Step
6: encode x_i into $p_\phi(z_i \mid x_i)$
7: split z_i in z_0 and z_1
8: decode z_0 and z_1 to obtain $p_\tau(x_i \mid z_0, z_1)$
9: decode z_1 to obtain $p_\theta(y_i \mid z_1)$
10: decode z_0 to obtain $p_\delta(y_i \mid z_0)$
11:
12: update ϕ, θ, τ by taking a gradient step
13:
14: Maximisation Step
15: encode x_i into $p_\phi(z_i \mid x_i)$
16: split z_i in z_0 and z_1
17: decode z_0 to obtain $p_\delta(y_i \mid z_0)$
18:
19: Relaxing Gaussian Assumption
20: decode y from $h_\delta^{-1}(h_\delta(y))$
21:
22: update δ by taking a gradient step
23: **end for**
24: increase λ by factor l
25: **end for**

5.3.3 Training Procedure

Our learning procedure is described in Algorithm 2. For every epoch, we sample i minibatches, where the number of i is determined by the batchsize. In the first part of our training algorithm (see Minimisation Step), we try to minimise the mutual information between Z_0 and Y. Therefore, we first encode X in our latent Z (line 6). Subsequently, we split Z in Z_0 and Z_1. After, we decode Z to X and Z_1, Z_0 to Y in lines 6-10, respectively. Consequently, we update the parameters in line 12 by employing the loss function in Eq. 5.3.2. Having fixed the parameters δ we update the latent representation Z_0 such that it encodes minimal mutual information about Y.

The second step (see Maximisation Step) denotes the adversarial part of our model. Again

we encode X into Z and partition the space to Z_0 and Z_1. In contrast to the minimisation step, we try to predict Y from Z_0 by updating parameters δ while fixing the remaining neural network parameters ϕ, θ and τ (see Eq. 5.3.3). In order to relax the Gaussian assumption, we extend our model by two additional neural networks h and h^{-1} and add their parameters to the existing parameters δ in line 20.

As a last step, we increase the compression parameter λ by a predefined factor l (line 24) after every epoch. Finally, we can train the described adversarial algorithm by any gradient descent method until convergence.

5.4 Experiments

A detailed description about the setups as well as additional experiments can be found in the supplementary materials.

5.4.1 Artificial Dataset

Dataset.

Our dataset is generated as follows: Our input consists of two input vectors x_0 and x_1. Here, the input vectors are drawn from a uniform distribution defined on $[0,1]$ and further multiplied by 8 and subtracted by 4. All input vectors form our input matrix X. Subsequently, we define two latent variables z_0 and z_1. Here, z_0 and z_1 are calculated as $2x_0 + x_1 + 10^{-1}\epsilon$ where $\epsilon \sim \mathcal{N}(0,1)$. Last, we calculate two target variables y_0 and y_1. In doing so, y_0 is calculated by $5 \cdot 10^{-2}(z_0 + bz_0)\cos(z_0) + 2 \cdot 10^{-1}\epsilon_1$ where $\epsilon_1 \sim \mathcal{N}(0,1)$ and b is 10. y_1 is defined as $5 \cdot 10^{-2}(z_0 + bz_1)\sin(z_0) + 2 \cdot 10^{-1}\epsilon_2$ with $\epsilon_2 \sim \mathcal{N}(0,1)$. Thus, y_0 and y_1 form a spiral where the angle and the radius are highly correlated. For visualisation purposes, we bin and colour code the values of Y in the following experiments. During the training and testing phase, samples are drawn from this data generation process.

Experimental Setup:

STIB For our setup, our encoding network consists of two fully connected layers with 256 neurons without bias. This is followed by a latent layer with three nodes that models the means of our three dimensional latent space where the variances are modeled as free parameters. Here, we split Z in a one-dimensional space Z_1 and two-dimensional space Z_0 which contain no information about Y. The decoder uses two neural networks with fully connected layers with 256 neurons for reconstructing the input and predicting the target. In addition, we model an adversarial network with two fully connected layers each with 256 neurons to predict the target from our latent space Z_0. Since we use adversarial training, we define two Adam optimisers Kingma & Ba (2015) with a learning rate of 0.0001 and a batch size of 60 that optimise our objective on an alternating basis. For better visualisation, we discretise X and Y into 10 bins to colour-code to show the invariant parts of the data.

STIB without adversary For the STIB without adversary setup, we use exactly the same configuration as in the STIB setup. The only difference is that we remove the adversarial mutual information regulariser. Thus, we define only one Adam optimiser with a learning rate of 0.0001 and a batch size of 60 which optimises our loss function.

VAE This VAE setup uses also the same configuration as STIB. Similar to STIB without adversary, we skip the mutual information regulariser. In addition, we only define a shared latent space

Z to reconstruct X and Y in contrast to STIB. As an optimiser, we employ Adam with a learning rate of 0.0001 and a batch size of 60.

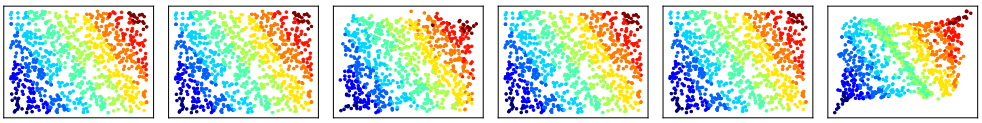
CVAE For CVAE, we use the same setup for encoder and decoder as in STIB. In contrast, we model the latent space Z with two dimensions by estimating the parameters μ and σ. We employ a two-dimensional latent space as we concatenate the one-dimensional part in traditional CVAE fashion. In order to train our model, we use Adam with a learning rate of 0.0001 and a batch size of 60.

CVIB For the last model, we use the equivalent model as in CVAE. In addition, we employ the mutual information regulariser which is described in Moyer et al. (2018).

Experiment 1. Reconstruction Capability

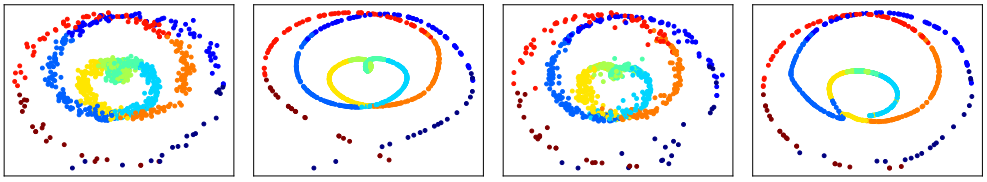
In the first experiment, we qualitatively inspect the ability of the latent space to approximately reconstruct X and Y from our latent representation. The reconstruction of X is depicted in Figure 5.4 and the reconstruction of Y in Figure 5.5. The color coded lines in the Figures 5.4 and 5.5 indicate the invariant parts in the dataset. Note in our artificial examples the invariances are continuous. However, we discretised the invariances into 10 colour-coded bins for visualisation proposes. In the first part, we compare our input X in Fig. 5.4a with the reconstruction ability of the models, namely STIB (Fig. 5.4b), VAE (Fig. 5.4c), STIB without adversary (Fig. 5.4d), CVAE (Fig. 5.4e) and CVIB (Fig. 5.4f). First, we inspect the ability to reconstruct the input X. From a visual perspective STIB, VAE, STIB without adversary, CVAE are approximately able to reconstruct the input X. Except CVIB can only partially reconstruct X. These qualitative support the quantitative findings which we have obtained in Experiment 2 in the main paper. In this experiment, we investigated the reconstruction of X by quantitatively evaluating the MAE.

In the second part, we examine the reconstruction capability of Y (Fig. 5.5). Here, we do not compare to CVAE and CVIB because in these particular models Y is used as an input and not reconstructed by the model. In comparison to the ground truth (Fig. 5.5a), STIB in Figure 5.5b is able to reconstruct the backbone of the spiral. We cannot reconstruct the Y in detail, because we draw noisy data points of Y (for more details see Dataset). In contrast, VAE (see Fig 5.5c) uses three latent dimensions Z instead of one. Therefore, VAE not only tries to reconstruct the spiral but also tries to learn the noise of the data. This is clearly indicated by the noisy reconstruction of the Y. Last, we compare to STIB without adversary in Figure 5.5d. Similarly to STIB, we are able to reconstruct Y which also confirms the quantitative findings from Experiment 2 in the main paper.



(a) (b) (c) (d) (e) (f)

Figure 5.4: The first image denotes the input X (Fig. 5.4a) whereas the second image (Fig. 5.4b) illustrates the reconstruction of STIB. Images three (Fig 5.4c) and four (Fig. 5.4d) denotes the reconstruction results of VAE and STIB without adversary, respectively. In Figure 5.4e, we show the reconstruction of CVAE and in Figure 5.4f the results of CVIB. For better visualisation, we discretise X into 10 bins to colour-code to show which part of the data is invariant.

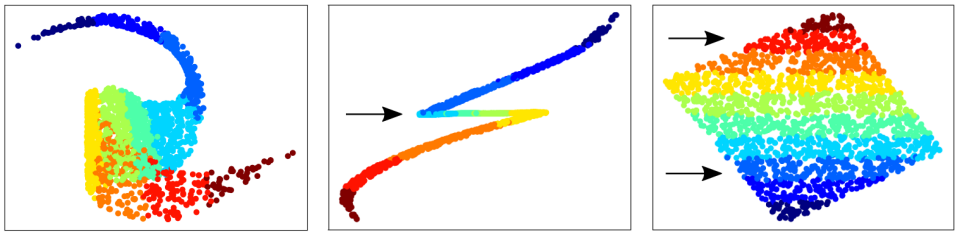


(a) (b) (c) (d)

Figure 5.5: The first image illustrates the output Y (Fig. 5.5a). The second image (Fig. 5.5b), illustrates the reconstruction results of STIB whereas the third column shows the VAE reconstruction of Y (Fig. 5.5c). The last column (Fig. 5.5d) shows the results for STIB without mutual information regulariser. We have not included results for CVAE and CVIB because Y is not reconstructed but used as an additional input. To better showcase which parts of the data is invariant, we discretise Y into 10 colour-coded bins.

Experiment 2. Examining the Latent Space

We demonstrate the ability of our model to learn a symmetry transformation which admits an analytical form from observations only. We compare our method with VAE (Gomez-Bombarelli et al., 2018) and STIB without regulariser for this purpose. Here, we use the same latent space structure that was described in the experimental setup. Subsequently, we plot the first dimension of Z_0 (x-axis) against Z_1 (y-axis) for all three methods. Due to the fact that every dimension in the VAE model contains information about the target, we plotted the first against the second dimension for simplicity. The horizontal coloured lines indicate that our approach (Fig. 5.6c) is able to learn an well defined symmetry transformation, because changing the value of the x-axis does not change the target Y. In contrast, the VAE (Fig. 5.6a) and STIB without any regulariser (Fig. 5.6b) are not able to preserve this invariance property and encode information about Y in Z_0 simultaneously. This can be clearly noted by the colour change of horizontal lines. That is, modifying the invariant space would still result in a change of Y and thus requires our mutual information regulariser.



(a) (b) (c)

Figure 5.6: Figure 5.6a depicts the latent space of VAE where the first two dimensions are plotted. In contrast, Figure 5.6b shows the latent space of STIB that was trained without our regulariser. Here, the invariant dimension Z_0 (x-axis) is plotted against the first dimension of Z_1 (y-axis). Figure 5.6c illustrates first dimension of the invariant latent space Z_0 (x-axis) plotted against Z_1 (y-axis) after being trained by our method. Horizontal coloured lines in the bottom right panel indicate invariant with respect to the target Y. In remaining panels the stripe structure is broken. Black arrows denote the invariant direction.

Experiment 3. Quantitative Evaluation

Here, we provide a quantitative comparison study with five different models in order to demonstrate the impact of our novel model architecture and mutual information regulariser. In addition to the models considered in Experiment 1, we compare to conditional VAE (CVAE) (Sohn et al., 2015) and conditional Information Bottleneck (CVIB) (Moyer et al., 2018) in Table 5.1. The setup is identical as described in the experimental setup. We compare the reconstruction MAE of X and Y as well as the amount of information which is remaining in the invariant space by measuring the mutual information between Z_0 and Y. Our study shows that we are able to obtain competitive reconstruction results for both X and Y for all of the models. However, we encounter a large difference between the models with respect to the remaining target information Y in the latent space Z_0. In order to quantify the differences, we calculated the mutual information using a nonparametric Kraskov estimator (Kraskov et al., 2004) to obtain a consistent estimate for all the models we compared. We specifically avoid using the Gaussian mutual information in our comparisons here, because in the other models the (non-transformed) Y is not necessarily Gaussian. Otherwise, we would end up with inaccurate mutual information estimates that make fair comparison infeasible. By using the Kraskov estimator, we observe that all competing models, Z_0 contain a large amount of mutual information about Y. In the VAE case, we obtain a mutual information if 3.89 bits and with our method without regularisation a value of 3.85 bits. Moreover, CVAE and CVIB still contain 2.57 bits and 2.44 bits mutual information, respectively. However, if we employ our adversarial regulariser, we are able to decrease the mutual information to 0.25 bits. That is, we have approximately removed all information about Y from Z_0. These quantitative results showcase the effectiveness of our method and support the qualitative results illustrated in Figure 5.6.

Table 5.1: Quantitative summary of results from artificial data. Here, we consider the VAE, STIB without regularization, CVAE, CVIB, STIB. For all models the MAE reconstruction errors for X and Y are considered as well as the mutual information (MI) in bits between the invariant space Z_0 and Y based on a Kraskov estimator. Lower MAE and MI is better. STIB outperforms each of the baselines considered.

MODEL	ARTIFICIAL EXPERIMENT		
	MAE(X)	MAE(Y)	$MI_K(Z_0,Y)$
VAE	0.21	**0.48**	3.89
STIB W/O ADV.	**0.01**	0.65	3.85
CVAE	0.33	-	2.57
CVIB	0.66	-	2.44
STIB	0.04	**0.47**	**0.25**

5.4.2 QM9 Dataset

Dataset.

We use the 134K organic molecules from the QM9 database (Ramakrishnan et al., 2014), which consists of up to nine main group atoms (C, O, N and F), not counting hydrogens. The chemical space of QM9 is based on the work of GDB-17 (Ruddigkeit et al., 2012), as the largest virtual database of compounds to date, enumerating 166.4 billion molecules of up to 17 atoms of C, N,

O, S, and halogens. GDB-17 is systematically generated using molecular connectivity graphs, and represents an attempt of complete and unbiased enumeration of the space of chemical compounds with small and stable organic molecules. Each molecule in QM9 has corresponding geometric, energetic, electronic and thermodynamic properties that are computed from Density Functional Theory (DFT) calculations. In all our experiments, we focus on a subset of this dataset with a fixed stoichiometry ($C_7O_2H_{10}$), consisting of 6095 molecules and corresponding bandgap energy and polarisability as invariant properties. By restricting chemical space to fixed atomic composition we can focus on how changes in chemical bonds govern these properties.

Experimental Setup:

STIB As input X we use the SMILES representation of a molecule, which encodes molecular connectivity in a string based graph and chemical properties as target Y. In doing so, we are converting the SMILES in a one-hot grammar representation based on the Grammar VAE introduced by Kusner *et al.* (Kusner et al., 2017). As proposed in Kusner *et al.* (Kusner et al., 2017), our encoder network consists of three 1D convolutional layers with 12 convolutional filters and a filter length of 3, followed by a 256 dimensional fully connected layer. In addition, we have two decoders that try to reconstruct both the chemical properties and the SMILES. The SMILES decoder consists of a 36 dimensional fully connected layer followed by three 256 dimensional Gated Recurrent Unit (GRU) layers. The properties decoder has four fully connected layers with 56, 256, 128, 2 nodes, respectively which is similar to Gomez-Bombarelli et al. (2018). Last, we define the adversarial network to minimize the mutual information between Z_0 and Y with the same configuration as the property decoder. Our model is trained using two Adam optimisers Kingma & Ba (2015) with an initial learning rate of 0.01 and a batch size of 36. Subsequently, we set our latent dimension Z to 16 because the reconstruction accuracy saturates (see Fig. 5 main paper). We split Z to Z_0 with 14 and Z_1 with 2 dimensions because we predict two target properties. To speedup the training procedure, we pretrained the encoder and decoder on approximately 100k molecules from the QM9 dataset and subsequently fine-tuned the latent representation on the fixed stoichiometry ($C_7O_2H_{10}$). All handling of chemistry was done with a Python interface to MOPACJames J. P. Stewart (2016) and GaussianFrisch et al., with use of RDKitrdk (2019) and ASELarsen et al. (2017).

STIB without adversary Here, we use the same configuration as in the STIB setup. However, we omit the adversary with the mutual information regulariser. As we do not have an adversary, our model uses only one Adam optimiser with an initial learning rate of 0.01 and a batch size of 36.

VAE In the VAE setup, we employ the equivalent architecture as in STIB without adversary. However, we do not partition Z into two separate latent spaces. Instead, we reconstruct X and Y from the Z directly. In this case, we devote Adam with an initial learning rate of 0.01 and a batch size of 36.

CVAE For the CVAE setup, we take the both encoder and decoder architecture of VAE. However, we do no reconstruct Y but concatenate Y with the latent representation Z in order to reconstruct X. Thus, our latent space has only a latent dimensionality of 14 instead of 16 in the VAE architecture. Similar to the models described before, we train CVAE using the Adam optimiser with an initial learning rate of 0.01 and a batch size of 36.

CVIB For CVIB, we use exactly the same setup as for CVAE. The only difference is that we add the mutual information regulariser, developed in Moyer et al. (2018) to the CVAE loss function.

Experiment 4. Model Selection

We inspect the molecule reconstruction ability of the input X given a varying number of latent dimensions (Fig. 5.7). To do so, we train our model on 95% of our dataset and subsequently evaluate on the remaining 5%. The model selection is hence performed by inspecting the reconstruction accuracy to select the optimal number of latent dimensions. In our case, the optimal model converges at 16 latent dimensions. Reconstructing molecules from lower dimensions is in general more challenging because there is a large number of molecules with similar bandgap energies and polarisability. This results in collisions which makes it difficult to resolve the many-to-one mapping in the latent space. In addition, we calculated the mutual information between Z_0 and Y using the Kraskov estimator. It is important to note that our model does not come with a trade-off between the reconstruction accuracy and being invariant against Y in Z_0. This property is clearly indicated in Figure 5.7 (blue line). Here, it can be observed that the mutual information constantly stays between 0.03 and 0.1 for all numbers of latent dimensions considered.

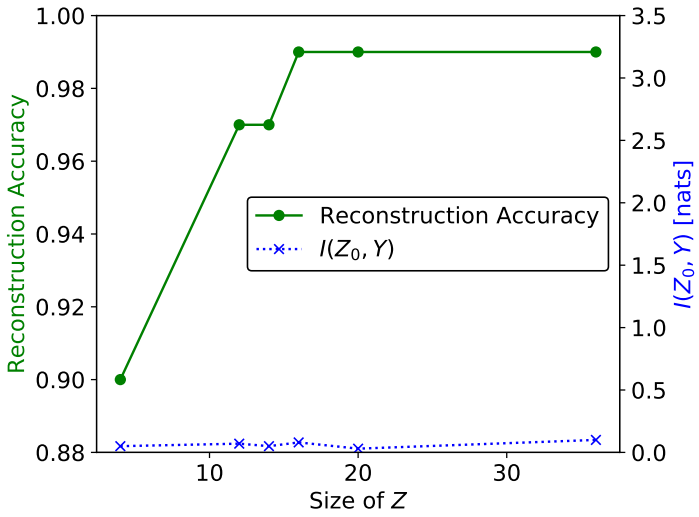
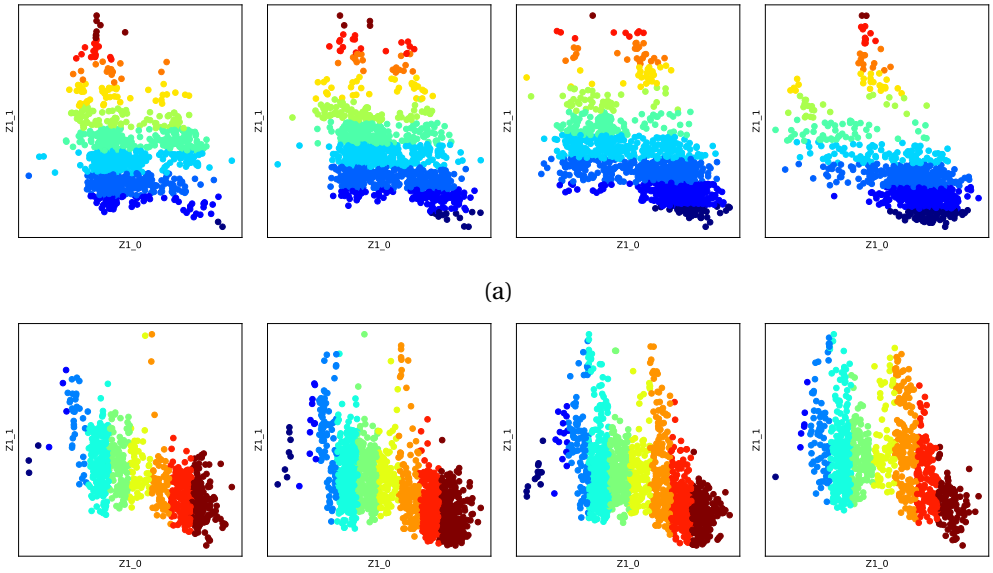


Figure 5.7: Illustration of the model selection process of STIB on the testset defined in Experiment 4. Therefore, the SMILES reconstruction accuracy (green dot) is considered. The x-axis denotes the number of latent dimensions whereas the left y-axis depicts the reconstruction accuracy of the molecules. The plot indicates that our reconstruction rate saturates at a level of 99% even when varying the number of latent dimensions. In addition, we plotted the mutual information (blue cross) between Z_0 and Y for all models which is depicted by the right y-axis.

Experiment 5. Examining the Latent Space

Here, we demonstrate that our model can generalise to more than one target (see Experiment 1), meaning novel materials have to satisfy multiple properties and at the same time have a structural invariant subspace. We train a model with a subspace (Z_0) which contains no information about the material depend properties, bandgap energy and polarisability. In order to illustrate this relationship, we plot the first two dimensions of the property space Z_1 and colour coded points according to intervals for bandgap energy and polarisability (Figure 5.8a and Figure 5.8b respectively). The color bins are equally spaced by the property range, where the minimum is

1.02 eV / 6.31 bohr^3 and the maximum is 16.93 eV / 143.43 bohr^3 for bandgap energies and polarisability, respectively. For simplicity and readability the we divide the invariant latent space Z_0 into ten sections and cumulatively group the points. Four sections were chosen for Figure 5.8. We note that binning is not necessary, but increases the readability of the Figure. In every Z_0-section, we observe the stripe structure which means that Z_0 is invariant according to the target. In contrast, if Z_0 would encode any property information, the stripe structure would be broken as demonstrated in panel 5.6a and panel 5.6b. Thus, our experiment clearly indicates there is no change in the latent space structure according to bandgap energy and polarisability. That is, exploring Z_0 will not affect the properties of the molecule.



(a)

(b)

Figure 5.8: Latent space plots for the first two dimensions in property dependent Z_1. Colours illustrates binned target properties, bandgap energies (Fig. 5.8a) and polarisabilities (Fig. 5.8b). The bins are equally spaced by the property range. The values lie between 1.02 eV / 6.31 bohr^3 and 16.93 eV / 143.43 bohr^3 for bandgap energies and polarisability, respectively. The four figures for each property denote four binned sections along the property invariant dimension Z_0, out of a total of ten sections. The invariance is illustrated by the lack of overlap of the colour patterns for each section in Z_0.

Experiment 6. Quantitative Evaluation

In this experiment, we perform a quantitative study on real data to demonstrate the effectiveness of our approach. We compare the baseline VAE, CVAE, CVIB, STIB without mutual information regularization of the latent space and STIB with mutal information regularization (Table 5.2). If we compare both the accuracy of correctly reconstructed SMILES strings and the MAE of the bandgap energy and polarisability, we obtain competitive reconstruction results. For all models considered in the quantitative study we received a SMILES reconstruction accuracy of 98% for VAE, 98%, for STIB without adversarial training scheme, 91% for CVAE, 76% for CVIB and 98% for STIB. In addition, the bandgap and polarisability MAE for the VAE is 0.28 eV and 0.75 bohr^3, respectively. In comparison, the STIB without adversary receives a bandgap error of 0.28 eV and

a polarisability error of 0.70 bohr^3. Moreover, STIB obtains a MAE for bandgap energy of 0.27 eV and 0.77 bohr^3 for polarisability. This shows that our approach receives competitive results in both reconstruction tasks in comparison to the baseline. As previously described in Experiment 3, we additionally calculated the mutual information with a Kraskov estimator between the target-invariant space Z_0 and the target Y. In order to get a better estimate, we estimated the mutual information on the whole dataset. For both the baseline and our model without regularisation, we received a mutual information on 1.54 bits and 0.66 bits, respectively. Here, 1.54 bits represent the amount of mutual information if the entire Z is considered (e.g. VAE). In addition, CVAE contains 0.56 bits mutual information. That implies that Z_0 still contains half the information about Y whereas if we employ our regulariser the mutual information is 0.09 bits. These quantitative results showcase that only STIB is able to remove all property information from Z_0 for real world applications and support the qualitative results obtained in Experiment 6. Solely, CVIB received a slightly better MI estimate with 0.03 bits, however it posses a vastly weaker reconstruction accuracy.

Table 5.2: Summary of quantitative results for QM9 experiments. Here, we consider VAE, STIB without regularization, CVAE, CVIB and STIB. The accuracy for SMILES and MAE reconstruction errors bandgap energy (eV), polarizability (bohr^3) are computed, as well as the mutual information (bits) between the invariant space Z_0 and Y based on a Kraskov estimator ($MI_K(Z_0,Y)$). Higher SMILES accuracy and lower MAE and MI are better. STIB outperforms the other baselines.

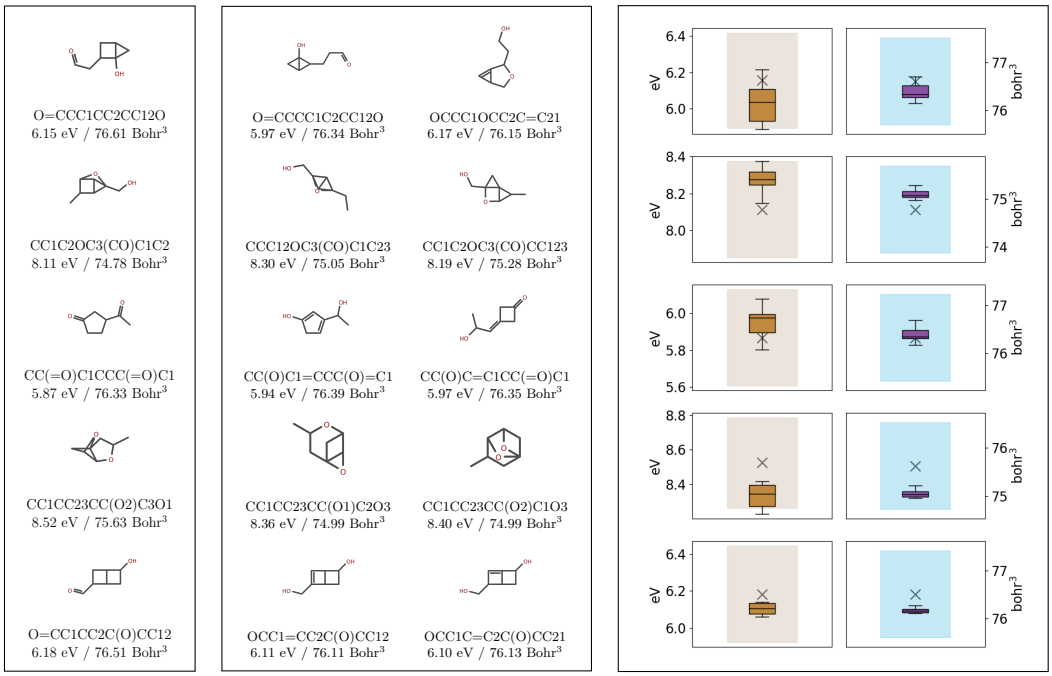
MODEL	QM9			
	SMILES	BANDGAP	POLARISABILITY	$MI_K(Z_0,Y)$
VAE	**0.98**	0.28	0.75	1.54
STIB W/O ADV.	**0.98**	0.28	**0.70**	0.66
CONDVAE	0.91	-	-	0.56
CVIB	0.76	-	-	**0.03**
STIB	**0.98**	**0.27**	0.77	0.09

Experiment 7: Evaluating the Generative Model

Here we investigate the generative nature of the model latent space and the consistency of property predictions. To do so, we explore five different points in property-latent space, according to five different reference molecules, as seen in Figure 5.9a. The points in property-latent spaces represent bandgap energies and polarizabilities of 6.15 eV / 76.61 bohr^3, 8.10 eV / 74.78 bohr^3, 5.87 eV / 76.33 bohr^3, 8.52 eV / 75.63 bohr^3, and 8.52 eV / 76.51 bohr^3, for rows one to five in Figure 5.9a respectively. Subsequently, from the property-irrelevant space, randomly sampled novel molecules are generated to SMILES, within the same stoichiometry and filtered as defined earlier (SMILES validity). The two generated molecules with the shortest latent space distance to the reference molecules are selected and depicted in Figure 5.9b.

We note that the nature of generative models is based upon chemical space defined by SMILES syntax. Therefore the model inherently learns how to generate correct chemistry purely from a SMILES syntax point-of-view. Training on the SMILES syntaxs will inherently result in a model which has a non-zero probability of generating unrealistic, high-energetic molecules which are not stable in nature, despite having a valid SMILES representation.

The self-consistency of the properties of the generated molecules are checked by using the model to find the position in property-latent space based upon the generated SMILES. Here, the predicted properties are within the expected error range (Table 5.2), which indicates the target-irrelevant consistency of our model. The error range is illustrated in Figure 5.9c) and compared to the expected error range (shaded boxes) of the model. The predicted properties averaged over all generated molecules from the five reference points (9-16 molecules per point) are within calculated the error range in Table 5.2. This experiment demonstrates the generative capabilities of the network by generating chemically correct novel molecules, which has self-consistent properties and is target-irrelevant within the expected error of the model.



(a) (b) (c)

Figure 5.9: Illustrating the generative process of our model. The experiment for five different molecules are located row-wise. Figure 5.9a show the reference molecules which serve as our starting point with their corresponding properties. In Figure 5.9b, we plotted two generated molecules which are closest to the reference molecule. The properties from the generated molecules are generated by using the sample molecules in the model and locating their corresponding position in property latent space (e.i. predicting the property). Additionally, we predict the properties of all molecules (9-16 per point) generated from our reference point and depict them as a box plot in Figure 5.9c, where the left box plot denotes the band gap energy and the right box plot the polarisability. The cross illustrates the true property of our starting point and the shaded background is the error confidence interval of our model.

Experiment 8: Quantum-Chemical Evaluation

In this experiment, we explore the generative aspect of our model in the context of accurately predicting the quantum properties for the sampled molecules. For this reason, we validated the properties associated with the generated compounds from first reference point (Figure 5.9b) using quantum chemical calculations. The properties for two molecules closets to the reference

molecule, using the same computational approach used in the generation of the QM9 dataset Ramakrishnan et al. (2014), were evaluated as a proof of concept. As described in Experiment 8, we use the first reference molecule in Figure 5.9 as a starting point in latent space, and generate compounds adjacent to the point. The two closest generated compounds were chosen for evaluation. From the generated compounds, a conformational scan was made, and the lowest confirmation was chosen for quantum chemical property evaluation, as according to pipeline documented by the reference dataset QM9. The property calculations of the generated molecules yield bandgap energy of 5.85 and 5.96 eV, respectively, which is within the error estimate of the model (Table 5.2) of the reference molecule (6.15 eV). In addition, the calculated polarisability is 76.39 and 77.72 bohr^3, which is just outside the estimated error space from the reference molecule (76.61 bohr^3). We acknowledge that the model is trained upon a graph representation of the molecule and therefore, no information regarding the property dependency on conformational changes are encoded. The choice of conformation is based upon the same pipeline used in generating the original dataset. However, the choice of conformation and structural optimization still has a random element to it and will, therefore, introduce another level of error going from SMILES to prediction. With that in mind, our model performs within the expected errors associated with the choices of approximations.

SMILES	Predicted		Calculated	
	Bandgap	Polarisability	Bandgap	Polarisability
O=CCC1CC2CC12O	NA	NA	6.15	76.61
O=CCCC1C2CC12O	5.97	76.34	5.85	76.39
OCCC1OCC2C=C21	6.17	76.15	5.97	77.72

Table 5.3: In this table, we evaluate the generated molecules. The first column denotes the SMILES string of the molecule and the second column describes the predicted bandgap energy (eV) and polarisability (bohr^3) which are estimated by our model. The third column denotes both the calculated bandgap energy and polarisability which are obtained with the same computational procedure as used to generate the reference dataset. The first row denotes the reference point of our generated novel molecules. Second and third row represent the closest novel molecules to the reference point and are generated by our model.

5.4.3 Zinc Dataset

Dataset.

In the third experiment, we use the 250K drug-like molecules from the ZINC database Gomez-Bombarelli et al. (2018). In contrast to QM9, this dataset consists of up to 23 heavy-atoms (C, O, N and F), not including hydrogens and offers a larger variety of molecule structures. The dataset is a randomly picked subset of the larger ZINC database Sterling & Irwin (2015) which contains over 17 million molecules. Here, every molecule has calculated drug-specific properties such as synthetic accessibility score(SAS) or the Qualitative Estimate of Drug-likeness (QED).

Experimental Setup:

STIB As input X we use the SMILES representation of a molecule as in the QM9 experiment. In contrast to the previous experiment, we are converting the SMILES in a one-hot representation based on DeepSMILES O'Boyle & Dalke (2018) instead of the grammar representation by Kusner et al. Kusner et al. (2017). DeepSMILES preprocesses a given SMILES string to a simpler representation which can be easier learned by recurrent neural networks. For our encoder network, we use one GRU layer with hidden 12 dimensions followed by two fully connected layers with 1356 and 128 dimensions, respectively. Subsequently, we set our latent dimension Z to 20 and split it to Z_0 with 19 and Z_1 with 1 dimension because we predict only the drug-likeliness of the molecules. Last, we define the decoder networks. Therefore, we have a SMILES decoder which consists of a 36 dimensional fully connected layer followed by three 501 dimensional Gated Recurrent Unit (GRU) layers. In addition, we define a property decoder which consists of four fully connected layers with 36, 36 and 1 nodes, respectively. The adversary decoder, which is responsible to minimise the mutual information between Z_0 and Y has got three layers with 36, 36 and 1 nodes, respectively. At the end, the model is trained using the Adam optimizer with an initial learning rate of 0.001 and a batch size of 500.

STIB without adversary As stated in the two experiments before, we use the same architecture as in STIB. However, we leave out its adversarial part. Thus, we employ only one Adam optimiser with an initial learning rate of 0.001 and a batch size of 500.

VAE Here, we make use of the same encoder/decoder architecture as in STIB without adversary. However, instead of splitting Z into two separate latent spaces Z_0 and Z_1, we reconstruct X and Y from Z.

CVAE We employ the the same setup as for the VAE. However, we do predict Y from Z but concatenate Y with the latent representation Z. For this reason, we set the dimensionality of Z 14 dimensions in contrast to 16 dimensions in the VAE architecture. Again, we use Adam to train our model with an initial learning rate of 0.01 and a batch size of 36.

CVIB Here, we use the identical setup as for CVAE. The merely distinction is the mutual information regulariser introduced in Moyer et al. (2018) which is added to the CVAE loss function.

Experiment 9. Quantitative Evaluation

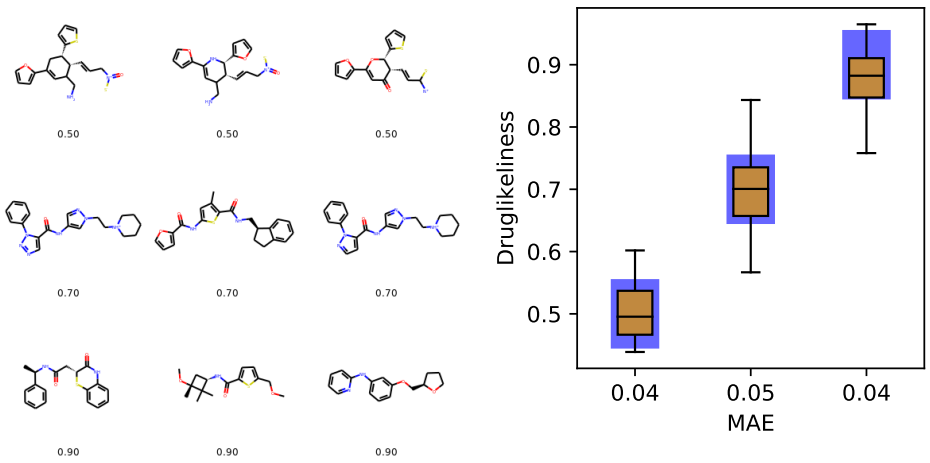
In this experiment, we perform an additional quantitative study on Zinc dataset (Table 5.4). Here, we also obain competitive reconstruction results in terms of SMILES accuracy and druglikeliness. For all models, we received a SMILES reconstruction accuracy of 98% for VAE, 98%, for STIB without adversarial training scheme, 98% for CVAE, 94% for CVIB and 98% for STIB. Furthermore, we investigated the druglikeliness MAE where all models received 0.05. This shows that our approach receives competitive results in both reconstruction tasks in comparison the baseline. Last, we estimated the mutual information with a Kraskov estimator between the target-invariant space Z_0 and the target Y. The VAE baseline contains 0.80 bits mutual information whereas STIB without adversary contains 0.24 bits. Moreover, we received a mutual information on 0.28 bits and 0.29 bits for CVAE and CVIB, respectively. That implies that all considered models contain mutual information in Z_0 about Y whereas if we employ STIB the mutual information is approximately eliminated (0.07 bits). These results confirm the findings of Experiment 2 and 5 that only STIB is able learn symmetry transformations from data while archiving competitive reconstruction results.

Table 5.4: Summary of quantitative results Zinc experiments. Here, we consider VAE, STIB without regularization, CVAE, CVIB and STIB. The accuracy for SMILES and MAE the reconstruction error for druglikeliness (probability) are computed, as well as the mutual information (bits) between the invariant space Z_0 and Y based on a Kraskov estimator ($MI_K(Z_0,Y)$). Higher SMILES accuracy and lower MAE and MI are better. STIB outperforms the other baselines.

MODEL	ZINC		
	SMILES	DRUGLIKELINESS	$MI_K(Z_0,Y)$
VAE	**0.98**	**0.05**	0.80
STIB W/O ADV.	**0.98**	**0.05**	0.24
CONDVAE	**0.98**	-	0.28
CVIB	0.94	-	0.29
STIB	**0.98**	**0.05**	**0.07**

Experiment 10. Generative Evaluation

Lastly, we investigate the generative nature and investigate the property consistency of our model. To do so, we fix three different points in property-latent space Z_1. The points in property-latent spaces represent a druglikeliness of 0.5, 0.7, 0.9, for rows one to three in Figure 5.10a, respectively. After, we randomly sample points in the invariant latent space Z_0 which are subsequently generated to SMILES strings.



(a) (b)

Figure 5.10: Illustration of the generative process of our model. Figure 5.10a shows samples drawn by our model. The labels represent the predicted druglikeliness properties which were estimated by out model. Each row in Figure 5.10a denotes molecules generated with a predefined druglikeliness. We further estimate the properties of the generated molecules and show the result in Figure 5.10b. The blue shaded background is the error confidence interval of our model and the x-axis denotes the MAE of all samples in the boxplot.

Having generated novel SMILES with potentially identical druglikeliness, we now perform a self-consistency check. That is, we feed the generated SMILES into our model and predict the

properties. If our model has learned an invariant representation the predicted druglikeliness should be identical to the fixed druglikeliness within the model error. We summarised the results of the model-consitency check in Figure 5.10b. Here, we plot the predicted druglikeliness averaged over all generated molecules from the three reference points using a boxplot. Every boxplot contains between 108 and 193 sampled molecules. The x-axis denotes the druglikeliness MAE whereas the blue box denotes the model error. The predicted properties averaged over all generated molecules from the three reference points posses a MAE between 0.04 and 0.05 which lies within calculated the model error range in Table 5.2. This observation is additionally supported by investigating the boxplots. Here, the predominant proportion of molecules lie within the model error range (blue box). Hence, this experiment demonstrates the generative capabilities of STIB by generating chemically correct novel molecules within the model's error range.

5.5 Conclusion

Symmetry transformations constitute a central building block for a large number of machine learning algorithms. In simple cases, symmetry transformations may be formulated analytically, e.g. for rotation or translation (Figure 5.1a). However, in many complex domains, such as the chemical space, invariances can only be observed from data, for example when different molecules have the same bandgap energy (Figure 5.1b). In the latter scenario, the corresponding symmetry transformation cannot be formulated analytically. Hence, learning such symmetry transformations from observed data remains a highly relevant yet challenging task, for instance in drug discovery. To address this task, we make three distinct contributions:

1. We have presented the *STIB* method, a novel generative model that is able to learn arbitrary symmetry transformations from observations alone via adversarial training and a partitioned latent space;
2. In addition to our modelling contribution, we provide a technical solution for continuous mutual information estimation based on correlation matrices;
3. Experiments on an artificial as well as two molecular datasets show that the proposed model learns symmetry transformations for both well-defined and arbitrary functions, and outperforms state-of-the-art methods on removing information from the invariant space up to factor 8.

Chapter 6

Learning Symmetry Transformations with Cycle-Consistent Regularisation

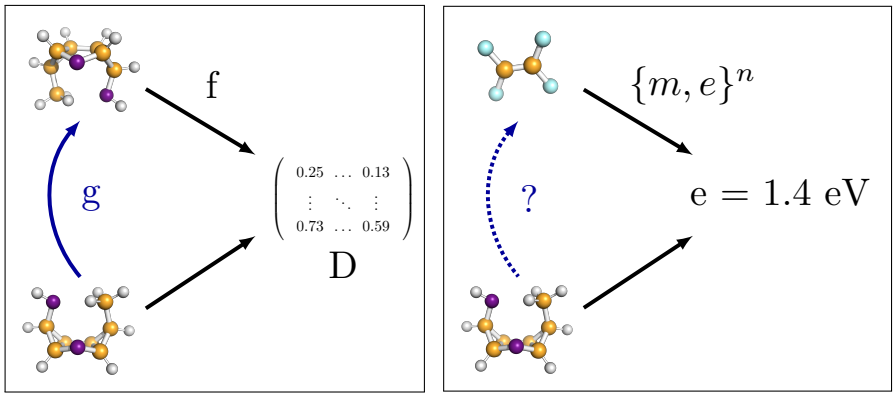
This chapter currently in submission at Neural Computation.

6.1 Introduction

In this chapter, we introduce a complementary approach to learn an known class of symmetry transformations g. Therefore, we reconsider our running example of rotation as a symmetry transformation where an illustration is depicted in Figure 6.1a. Here, we observe a 3D configuration of a molecule denoted as m. This molecule is in 3D space by applying the symmetry transformation g. For any rotation g, we calculate the distance matrix D between the atoms of the rotated molecule $g(m)$ with a predefined function f. In this simple case, rotation admits a straightforward analytical form. Due to the fact that the symmetry transformation g induces an invariance class, we obtain the same distance matrix for every rotation g, that is $f(m) = f(g(m))$ for any rotation g.

However symmetry transformation also exist in highly complex domains such as the chemical space. In such domains, analytical forms of symmetry transformations g are difficult or impossible to find (Figure 6.1b). A prevalent example, where we would like to find such a symmetry transformation, is the discovery novel molecules for the design of organic solar cells in material science. In this case, all molecules must possess a specific property, namely a bandgap energy of approximately 1.4 eV Shockley & Queisser (1961), in order to adequately generate electricity from the solar spectrum. In such scenarios, no predefined symmetry transformation (such as rotation) is known or can be assumed. The only available data which defines our invariance class are the $\{m, e\}^n$ numeric point-wise samples from the function f where n is the number of samples, m the molecule and $e = f(m)$ the bandgap energy. That is, we can observe such invariances from point-wise sample yet there exist no analytical form of a symmetry transformation g which alters the molecule m and leaves the bandgap energy e unchanged can be assumed.

Hence, the aim of our method is to learn the class of symmetry transformations g which result in a symmetry property f from unstructured data. Therefore, we learn a continuous data representation Z and the corresponding symmetry transformation g from data samples $\{m, e\}_1^n$ only. To this end, we propose the Cycle-Consistency Information Bottleneck (CCIB) where we transform the input X (e.g. a molecule) into a latent representation Z. Subsequently, we decode Z to X and a property Y (e.g. the bandgap energy). Specifically, we divide the latent space into



(a) Rotational symmetry transformation.

(b) Unknown symmetry transformation.

Figure 6.1: A molecule is rotated by g admitting an analytical form in Figure 6.1a. The distance matrix D between atoms is calculated by a known function f and remains unchanged for all rotations. In Figure 6.1b, n samples $\{m,e\}^n$ where m is the molecule and e the bandgap energy. These samples approximate the function f whereas the class of functions g leading to the same bandgap energy is unknown.

two subspaces Z_0 and Z_1 to explore the variations of the data with respect to a specific target. In our model, Z_1 is the subspace that contains information about the target variable Y, while Z_0 is the subspace that is invariant to the Y and contains additional information about X. In this way, we are able to separate the symmetry transformation g in the isolated latent space Z_0 such that is does not alter the target Y.

Unfortunately, a simple partitioning of the latent representations is not sufficient to prevent the encoder to leak information into the invariant space Z_0. Recent works have already focused on learning such symmetry transformations Klys et al. (2018); Lample et al. (2017); Wieser et al. (2020) while preventing leakage into Z_0. To do so, these methods employed adversarial training which removed target-information Y from the invariant space Z_0. Despite their ability to prevent the encoder to code target information about Y in Z_0, the aforementioned methods suffer from two limitations: First, the method employ adversarial training which can lead to convergence issues due to the alternating optimisation procedure Mescheder et al. (2018). Second, all methods are unable to deal with mixed data as they are restricted to either discrete or continuous target variables Y.

To overcome the aforementioned issues, we introduce a novel regularisation method based on cycle-consistent regularisation between the latent representation Z which serves as the core element of our method. To do so, we generate new latent representations Z by fixing Z_1 and varying the invariant space Z_0. Based on this representation, we sample new data which we map back to Z using the encoder of our model. If the encoder would have learned a unique mapping that is able to separate the information between Z_0 and Z_1, we should arrive at our initial position in Z. As this is usually not the case, we will penalise the encoder by minimising the loss of the initial and estimated position in Z. Due to the fact that our model minimises the loss between the latent representation, we are able to predict mixed discrete and continuous data, simultaneously. In addition, we add our new loss term as a penalty directly to the information bottleneck optimisation objective which prevents from the alternating optimisation objective in adversarial training.

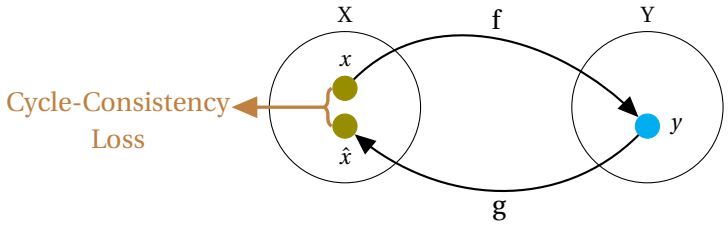


Figure 6.2: We illustrate the concept of cycle-consistency. A point x is mapped from domain X to domain Y using a function f. Subsequently, we employ a function g to map y back to $\hat{x}$ in domain X. The distance between x and $\hat{x}$ denotes the cycle-consistny loss.

In summary, we make the following contributions:

1. We introduce a deep information bottleneck model that learns a continuous low-dimensional representation from unstructured input data.

2. We further propose a cycle-consistent regularisation approach. This makes it possible to deal with mixed discrete and continuous data, simultaneously and overcome the convergence issues of adversarial training by incorporating the regulariser into the information bottleneck loss directly.

3. Our experiments on both artificial as well as molecular datasets showcase that our approach model learns both pre-defined and arbitrary symmetry transformations and outperforms state-of-the-art methods.

6.2 Cycle-Consistency

Here, we describe the core of our CCIB model to remove information which was encoded in Z_0. To do so, we introduce a cycle-consistent regularisation approach which maps a data point to its initial position after the data point was transferred to a different space. More precisely, consider the following setting where a domain X consists of summer landscapes whereas a second domain Y consists of winter landscapes. We employ a function $f(x)$ to transform a summer landscape x to the corresponding winter landscape y. Subsequently, we try to map y back to the domain X using a function $g(y)$ which should be approximately the same x called $\hat{x}$. A illustration of the described concept is visualised in Figure 6.2. In most cases, there exist a discrepancy between x and $\hat{x}$ which is called the cycle-consistency loss. In order to obtain an approximately invertible mapping we minimise the following loss function:

$$\mathscr{L}_{cycle} = \|(g(f(x)) - x\|_1 \tag{6.2.1}$$

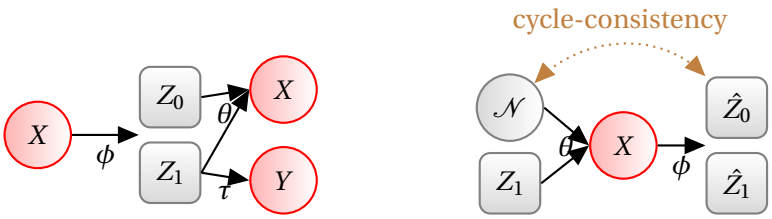
6.3 Cycle-Consistent Information Bottleneck

As previously described in Section 6.1, our goal is to learn symmetry transformations g based on observations X and Y (see Figure 6.1b). Here, X and Y may be complex objects, such as molecules and its corresponding bandgap energies, which are difficult to manipulate consistently. In order to overcome this issue, we aim to learn a continuous low-dimensional representation of our input data X and Y in Euclidian space. To do so, we augment the traditional

deep information bottleneck formulation (Eq. (4.2.1)) with an additional decoder reconstructing X from Z. Our base model is thus defined as an augmented parametric formulation of the information bottleneck (see Eq. (4.2.1)):

$$\min_{\phi,\theta,\tau} I_\phi(Z;X) - \lambda(I_{\phi,\theta}(Z;Y) + I_{\phi,\tau}(Z;X)), \tag{6.3.1}$$

where ϕ, θ, τ describe neural network parameters and λ the compression factor.



(a) (b)

Figure 6.3: Graphical illustration of our two-step adversarial training approach. Red circles denote observed input/output of our model. Gray rectangles represent the latent representation which is divided into two separate subspaces Z_0 and Z_1. Black arrows describe neural networks with trainable parameters. Greek letters define neural network parameters. In the first step (Figure 6.3a), we try to learn a representation of Z_0 which minimises the mutual information between Z_0 and Y by updating ϕ, θ and τ. In the second step (Figure 6.3b), we sampled randomly from Z_0 and try to map X to the sampe point in Z by employing a cycle-consistency loss.

Based on the model specified in Eq. (6.3.1), we provide a novel approach to learn symmetry transformations on latent representations Z. To this end, we propose partitioning the latent space Z into two components, Z_0 and Z_1. Z_1 is intended to capture all information about Y, while Z_0 should contain all remaining information about X. That is, changing Z_0 should change X but *not affect* the value of Y. Thus, Z_0 expresses a surrogate of the unknown function g which is depicted in Figure 5.1b. Therefore (see Figure 6.3a), our model learns a low-dimensional representation Z_0 and Z_1 of X and Y by maximising the mutual information between $I_{\phi,\tau}(Z_0, Z_1; X)$ and $I_{\phi,\theta}(Z_1; Y)$:

$$\mathscr{L}_1 = \min_{\phi,\theta,\tau} I_\phi(X;Z) - \lambda\Big(I_{\phi,\tau}(Z_0, Z_1; X) + I_{\phi,\theta}(Z_1; Y)\Big) \tag{6.3.2}$$

In practice, we calculate the mutual information by modelling $p(z \mid x)$, $p(x \mid z)$ and $p(y \mid z_1)$ using neural networks. To do so, we estimate z by learning a function $f_\phi(x)$ where x is a sample of the input. In order to reconstruct x and y, we learn functions $g_\theta(z)$ and $h_\tau(z_1)$, respectively. However, simply partitioning the latent space is not sufficient, since information about Y may still be encoded in Z_0. To address this task, we propose combining the model defined in Eq. (6.3.1) with a cycle-consistent regularisation approach.

6.3.1 Cycle-Consistent Regularisation

The central element of our approach is a cycle-consistency loss in the latent space. A sketch of the concept is illustrated in Figure 6.3b. The idea is to fix a point in Z_1 and sample a random point in the invariant space Z_0 which results in a point $\tilde{z}$ in the latent space. From this point in

the latent space, we generate a new data point $\hat{x}$ by using the previously defined function $g_\theta(\tilde{z})$. Subsequently, we map $\hat{x}$ to $\hat{z}$ by employing $f_\phi(\hat{x})$. In theory if the encoder is able to decompose the target information in Z_1 and the invariant information in Z_0, the estimated $\hat{Z}$ should map to map to the same position as the starting point $\tilde{Z}$. However in practise the encoder does not necessarily learn such a mapping. To overcome this limitation, we introduce a cycle-consistency loss which penalises the encoder if it maps $\hat{Z}$ to far away from the starting point $\tilde{Z}$:

$$\begin{aligned} \mathscr{L}_{cycle} &= \beta \| (f(h(\tilde{Z})) - \hat{Z} \|_1 \\ &= \beta \| (\tilde{Z} - \hat{Z} \|_1 \end{aligned} \tag{6.3.3}$$

The resulting model thus reduces to a Deep Information Bottleneck model with a partitioned latent space which is simultaneously regularised with help of a cycle-consistency loss. This ensures that Z_0 contains no information about Y. In more detail, our complete approach is formulated as follows:

$$\begin{aligned} \mathscr{L}_{complete} &= \mathscr{L}_1 + \beta \mathscr{L}_{cycle} \\ \mathscr{L}_{complete} &= I_\phi(X;Z) - \lambda \Big(I_{\phi,\tau}(Z_0, Z_1; X) + I_{\phi,\theta}(Z_1; Y) + \beta \| (\tilde{Z} - \hat{Z} \|_1 \Big) \end{aligned} \tag{6.3.4}$$

In theory, it is sufficient to define the cycle-consistency loss on the target space Z_1 only. However, we found that defining the cycle-consistency loss on the full latent space Z leads to a better stability and an increased model performance, overall. In addition, a direct consequence of our approach is that we are able to deal with mixed discrete and continuous targets due to the cycle-consistent loss between $\hat{Z}$ and $\tilde{Z}$. This combination was not feasible in current state-of-the-art methods (Klys et al., 2018; Wieser et al., 2020).

6.3.2 Implementation and Training Procedure

In this section, we describe our implementation and the corresponding training procedure in Algorithm 3. To do so, we sample i minibatches per epoch from our training dataset, where i is the number of batches determined by the batch size. In the first part of our training algorithm (see Deep Information Bottleneck), we optimise the information bottleneck objective function described in Eq. 6.3.2. Here, we first encode X in our latent Z (line 6) and after we partition the latent space Z into latent subspaces Z_0 and Z_1. Subsequently, we decode Z to X and Z_1 to Y in lines 6-9, respectively.

The second step (see Cycle-Consistency), we employ the cycle-consistent regularisation part of our model to force the encoder to partition this information between Z_0 and $Z-1$ correctly. Therefore, we fix the latent representation in Z_1 from each minibatch in line 12. Afterwards, we sample random points in Z_0 by sampling from the model prior Z_0. These newly generated in the latent space Z are feed in the decoder to obtain new data samples $\hat{x}$ (line 14). Subsequently, these data samples are encoded to the latent space again in line 15. In the last step, we update the model parameters ϕ, θ and τ by taking a gradient step using the loss function depicted in Eq. 6.3.4.

As a last step, we increase the compression parameter λ by a predefined factor l (line 24) after every epoch. Finally, we can train the described adversarial algorithm by any gradient descent method until convergence.

Algorithm 3 Cycle-Consistency Information Bottleneck

Input: input x, target y

for each epoch **do**
  sample i minibatches of x and y
  for each minibatch i **do**

    Deep Information Bottleneck
    encode x_i into $p_\phi(z_i \mid x_i)$
    split z_i in z_0 and z_1
    decode z_0 and z_1 to obtain $p_\tau(x_i \mid z_0, z_1)$
    decode z_1 to obtain $p_\theta(y_i \mid z_1)$

    Cycle-Consistency
    fix z_1
    sample z_0 randomly from prior $\mathcal{N}(0,1)$
    decode z_0 and z_1 to obtain $\hat{x}_i$
    encode $\hat{x}_i$ into $p_\phi(\hat{z}_i \mid \hat{x}_i)$

    update ϕ, θ, τ by taking a gradient step
  end for
  increase λ by factor l
end for

6.4 Experiments

In this section, we demonstrate the effectiveness of our method described in Section 6.3 on both artificial and molecular data.

6.4.1 Artificial Dataset

Dataset.

The input X of our artificial dataset consists of two input vectors x_0 and x_1. We draw the input vectors from a uniform distribution defined on $[0,1]$. We further scale the uniform samples as follows: x = (x*8)-4. These input vector x_0 and x_1 form our input matrix X. Subsequently, we define a latent variable z_1 which is defined as a scaled l1-norm plus Gaussian noise: $2x_0 + x_1 + 10^{-1}\epsilon$ where $\epsilon \sim \mathcal{N}(0,1)$. By using the l1-norm, every point that lies on the inverse diagonal of X is mapped to the same point in Z. Therefore, we need a second orthogonal space to reconstruct the correct position in X. Based on z_1, we calculate two target variables y_0 and y_1. In doing so, y_0 is calculated by $5 \cdot 10^{-2}(z_0 + bz_0)\cos(z_0) + 2 \cdot 10^{-1}\epsilon_1$ where $\epsilon_1 \sim \mathcal{N}(0,1)$ and b is 10. y_1 is defined as $5 \cdot 10^{-2}(z_0 + bz_1)\sin(z_0) + 2 \cdot 10^{-1}\epsilon_2$ with $\epsilon_2 \sim \mathcal{N}(0,1)$. Therefore, y_0 and y_1 form a spiral where the angle and the radius are highly correlated. We bin and colour code the values of Y in the following experiments for visualisation purposes. During the training and testing phase, samples are drawn from this data generation process.

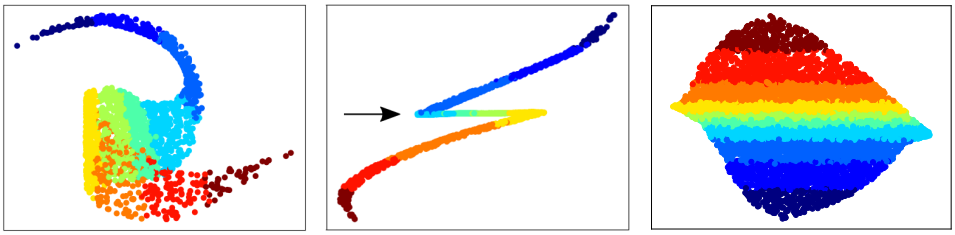
Experimental Setup.

Our model is implemented as follows: The encoder encoder consists of two fully connected layers with 256 neurons which is followed by a latent layer with three nodes. These nodes depict the

means of our three dimensional latent space. Instead of learning the variance for every single data point by the network, we model the global variance for each dimension by employing free parameters. In this case, we partition Z in a one-dimensional space Z_1 and two-dimensional space Z_0. Here, Z_0 should contain no information about Y. The decoder is implemented by two neural networks with fully connected layers with 256 neurons for reconstructing both the input and predicting the target. In order to train our model, we define an Adam optimiser (Kingma & Ba, 2015) with a learning rate of 0.0005 and a batch size of 60 which optimises our objective. For better visualisation, we discretise Y into 10 bins to colour-code the latent space.

Experiment 1. Examining the Latent Space

We showcase the ability of our model to learn a symmetry transformation which admits an analytical form by examining the latent space Z of our model. Therefore, we compare our method with VAE (Gomez-Bombarelli et al., 2018) with a split latent space for this purpose. To do so, we employ the equivalent latent space structure which has been previously described in the experimental setup. For a qualitative examination of the latent space, we plot the first dimension of Z_0 (x-axis) against Z_1 (y-axis) for all three methods. We plotted the first against the second dimension for simplicity as every dimension in the VAE model contains information about the target. The horizontal coloured lines indicate that our approach (Fig. 6.4c) is able to learn a well defined symmetry transformation due to the fact that varying the value of the x-axis does not alter the target Y. In contrast, both the VAE (Fig. 6.4a) and VAE with partitioned latent space (Fig. 6.4b) are not able to preserve this invariance property and thus store information about Y in Z_0 simultaneously. This claim can be validated by the colour change of horizontal lines. That is, altering the invariant space Z_0 would still result in a change of Y and thus requires our cycle-consistent regulariser.



(a) (b) (c)

Figure 6.4: Figure 6.4a illustrates the latent space of VAE where the first two dimensions are plotted. Figure 6.4b plots the latent space of VAE with a partitioned latent space. In this case, the first dimension of Z_0 (x-axis) is plotted against the invariant dimension of Z_1 (y-axis). In contrast, Figure 6.4c shows the first dimension of the invariant latent space Z_0 (x-axis) against Z_1 (y-axis) after being trained with our cycle-consistent regulariser. The horizontal coloured lines in the right panel indicate invariance with respect to the target Y. In remaining panels the stripe structure is broken which suggest target information in the invariant space Z_0. Black arrows denote the invariant direction.

Experiment 2. Quantitative Comparison

In the second experiment, we showcase a quantitative comparison study with six different models in order to demonstrate the impact of our Cycle-Consistent Information Bottleneck. In ad-

dition to the models considered in Experiment 1, we compare to conditional VAE (CVAE) (Sohn et al., 2015), Symmetry-Invariant Information Bottleneck (STIB) (Wieser et al., 2020) and conditional Information Bottleneck (CVIB) (Moyer et al., 2018) in Table 6.1. In our quantiative comparison, we investigate the reconstruction MAE of X and Y as well as the amount of mutual information between Z_0 and Y. Our study indicates that we are able to obtain competitive reconstruction results for both X and Y for all of the models. However, we obtain a significant difference between the models with respect to the remaining target information Y in the latent space Z_0. In order to quantify this differences, we estimated the mutual information using a nonparametric Kraskov estimator (Kraskov et al., 2004) to obtain a consistent estimate for all the models. We specifically avoid using the Gaussian mutual information in our comparisons here, because the random variables X and Y are not necessarily Gaussian. This would otherwise result in inaccurate mutual information estimates which makes fair comparison infeasible. By using the Kraskov estimator, we observe that for all competing models, Z_0 contains a large amount of mutual information about Y. Especially in the VAE case, we get a mutual information of 3.89 bits and for the VAE with a partitioned latent space a value of 3.85 bits. Furthermore, CVAE and CVIB also contain 2.57 bits and 2.44 bits mutual information, respectively. In addition, we compared our method to STIB where received a mutual information of 0.25 bits. In contrast, if we use our cycle-consistent regulariser, we are able to remove almost all mutual information from Z_0 (0.02 bits). These quantitative results demonstrate the effectiveness of our method and support the qualitative findings which have been illustrated in Figure 6.4.

Table 6.1: This table illustrates the quantitative results of the artificial experiment. In this experiment, we consider the VAE, VAE with partitioned latent space, CVAE, CVIB, STIB and CCIB. We compare the MAE reconstruction error for X and Y as well as the mutual information (MI) between Z_0 and Y based on a Kraskov estimator. A Lower MAE and MI is better which indicates that CCIB outperforms each of the baselines considered.

MODEL	ARTIFICIAL EXPERIMENT		
	MAE(X)	MAE(Y)	$MI_K(Z_0,Y)$
VAE	0.21	0.48	3.89
STIB W/O ADV.	**0.01**	0.65	3.85
CVAE	0.33	-	2.57
CVIB	0.66	-	2.44
STIB	0.04	0.47	0.25
CCIB	0.02	**0.43**	**0.02**

6.4.2 QM9 Dataset

Dataset.

We look at the QM9 database (Ramakrishnan et al., 2014) with 134K organic molecules, which consists of up to nine main group atoms (C, O, N and F), not counting hydrogens. The QM9 dataset is a subset of the much larger GDB-17 database (Ruddigkeit et al., 2012) which is the largest virtual database of compounds to date. This database consists of in total 166.4 billion molecules of up to 17 atoms of C, N, O, S, and halogens. In particular, GDB-17 is systematically generated using molecular connectivity graphs, and represents an attempt of complete and unbi-

ased enumeration of the space of chemical compounds with small and stable organic molecules. Every molecule in QM9 comes with its corresponding geometric, energetic, electronic and thermodynamic properties. These properties have been computed with help of Density Functional Theory (DFT) calculations. In all our experiments, we focus on a subset of this dataset with a fixed stoichiometry ($C_7O_2H_{10}$). This subset consists of 6093 molecules where we look at the bandgap energy as invariant property of our model.

Experimental Setup.

As input x we use the SMILES representation of a molecule as in the QM9 experiment. This representation describes the graph of a molecules based on the SMILES grammar. As learning SMILES representations with neural networks is challenging, we are converting the SMILES in a one-hot representation based on DeepSMILES (O'Boyle & Dalke, 2018). DeepSMILES preprocesses a given SMILES string to a simpler representation which can be easier learned by recurrent neural networks. For the encoder of our model, we employ three GRU layers with hidden 12 dimensions followed by one fully connected layers with 128 dimensions. Subsequently, we set our latent dimension Z to 10 and partition it to Z_0 with 9 and Z_1 with 1 dimension because we predict only the bandgap energy of the molecules. In the last part, we define the two decoders of our model. To this end, we have a SMILES decoder which consists of a 36 dimensional fully connected layer followed by three 501 dimensional Gated Recurrent Unit (GRU) layers. Moreover, we define a property decoder which consists of two fully connected layers with 36 and 1 nodes, respectively. At the end, the model is trained using the Adam optimizer with an initial learning rate of 0.0005 and a batch size of 60.

Experiment 3. Quantitative Comparision

We investigate in a quantitative study on molecular data the effectiveness of our approach. To do so, we compare the baseline VAE, CVAE, CVIB, VAE with partitioned latent space, STIB and our cycle-consistent regularisation approach (Table 6.2). If we compare both the accuracy of correctly reconstructed SMILES strings and the MAE of the bandgap energy, we obtain competitive reconstruction results. For all models considered in the quantitative study we received a SMILES reconstruction accuracy of 95% for VAE, 93%, for VAE with partitioned latent space, 95% for CVAE, 82% for CVIB and 94% for STIB and 93% for CCIB. Moreover, the bandgap MAE for the VAE is 0.22 eV . In contrast, the VAE with partitioned latent space receives a bandgap error of 0.23 eV as well as STIB which also obtains a MAE for bandgap energy of 0.23 eV. This shows that our approach receives competitive results in both reconstruction tasks in comparison to the baseline. As previously described in Experiment 2, we additionally calculated the mutual information with a Kraskov estimator between the target-invariant space Z_0 and the target Y. For both the VAE and the VAE with partitioned latent space, we received a mutual information on 1.11 bits and 0.53 bits, respectively. Here, 1.11 bits represent the amount of mutual information if the entire Z is considered (e.g. VAE). Furthermore, CVAE contains 0.53 bits and CVIB 0.40 bits mutual information. That implies that Z_0 still contains half the information about Y. In addition, STIB still holds an amount of 0.23 bits of mutual information in Z_0. In contrast, CCIB only contains 0.05 bits mutual information with simultaniously the best reconstruction results. These quantitative results showcase that only CCIB is able to remove approximatly all property information from Z_0 for real world applications.

Table 6.2: This table summarises the quantitative results for QM9 experiment. In this experiment, we compare VAE, VAE with partitioned latent space, CVAE, CVIB, STIB and CCIB. The accuracy for SMILES and MAE reconstruction errors bandgap energy (eV) are computed, as well as the mutual information (bits) between the invariant space Z_0 and Y based on a Kraskov estimator ($MI_K(Z_0,Y)$). Higher SMILES accuracy and lower MAE and MI are better. CCIB outperforms the other baselines.

MODEL	SMILES	BANDGAP	$MI_K(Z_0,Y)$
VAE	**0.95**	**0.22**	1.11
VAE W/O ADV.	0.93	0.23	0.53
CVAE	**0.95**	-	0.40
CVIB	0.82	-	0.37
STIB	0.94	0.23	0.23
CCIB	0.93	**0.22**	**0.05**

Without loss of generality, our method is also able to deal with discrete or mixed targets despite the fact that our experiments have shown only results for the continuous case.

6.5 Conclusion

Learning invariant representations are a crucial building block for various of machine learning methods for model understanding and target-specific exploration of the data. In basic cases, such symmetry transformations can be formulated analytically, such as for rotation in Figure 6.1a). In more complex domains, such as the chemical space, invariances can only be observed from unstructured data by point-wise samples. For example in chemistry, there exist multiple different molecules which have the same bandgap energy as described in Figure 6.1b. Unfortunately, the corresponding symmetry transformation for such cases cannot be formulated analytically. For this reason, learning such symmetry transformations from observed data remains a highly relevant yet challenging task in many application areas such as drug discovery or material design. To address this task, we make three distinct contributions:

1. We have presented the *CCIB* method, a novel deep information bottleneck model that is able to learn arbitrary symmetry transformations from observations alone by employing a partitioned latent space;
2. In order to prevent the encoder leak target information in the invariant space Z_0 we introduced a cycle-consistency loss in the latent space. This approach circumvents the problem associated with adversarial training and allows for mixed discrete and continuous data as target variables Y;
3. Our experiments on both artificial as well as molecular datasets showcase that our approach model learns both pre-defined and arbitrary symmetry transformations and outperforms state-of-the-art methods.

Chapter 7

Conclusion

In the last chapter, we recap our research questions and summarise the contributions of the thesis. Subsequently. we describe existing limitations of the proposed methods and give an outlook on future work.

7.1 Summary

In recent years, deep latent variable model became a crucial part of modern machine learning algorithms as they are able to both deal with unstructured data and capture non-linear dependencies. However, such models do not learn target-specific invariances implicitly which leads to problems in interpretation and control. For this reason, we want to be incorporate such target-specific invariances into the model directly. As a consequence, the goal of this thesis was to learn invariant representations in the context of deep latent variable models for the following two problem settings:

1. How can we find a symmetry transformation g when f and g admit an analytical form yet f is only described by point-wise samples?

2. How can we learn symmetry transformations g if g has an unknown form and f is described by point-wise samples?

We summarise our contributions for the research questions (1) and (2) below:

Learning Symmetries by Property Exploitation. In our first contribution, we aim to solve our first research question. Here, we look at the case where we want to learn symmetries if the invariant feature extractor f is the mutual information. In theory, mutual information admits an analytical form. However in practice, estimating mutual information is challenging and has hence to be approximated by learning an invariant feature extractor f. In this setting, we know that a symmetry transformation g of mutual information denotes the class of all monotone increasing transformations. To this end, we exploit the properties of mutual information to learn an invariant feature extractor f which is invariant against g.

We applied this concept to the deep information bottleneck model (Alemi et al., 2017) which suffers from that issue. In order to overcome this limitation, we proposed a copula transformation based on ranks that restored the invariance properties of mutual information. Our experiments illustrate that our method leads to reliable mutual information estimates and structured latent

representations on both artificial and real-world datasets.

Learning of Unknown Symmetry Transformations with Mutual Information Regularisation. In the second contribution, we answer research question (2). In this case, the symmetry transformation g is unknown however f is known but can only be observed by point-wise samples. We tackled this problem by using a deep information bottleneck approach in combination with adversarial training. To this end, we partitioned the latent space into two parts Z_0 and Z_1. The first part contained all information that is necessary to estimate the symmetry. The second part served as a surrogate for g. That is, varying this space will not affect the symmetry. However, merely splitting this space is not sufficient as information may leak in Z_1. To overcome this issue, we proposed a continuous mutual information regulariser based on correlation matrices with a bijective variable transformation. We trained this model in an adversarial fashion which allowed to eliminate all information about the symmetry from this subspace. We evaluated our method on both artificial and molecular datasets.

Learning of Unknown Symmetry Transformations using Cycle-Consistency. In the last contribution, we propose a complementary approach to solve research question (2). We built on the same deep information bottleneck approach as proposed in Chapter 5. In contrast, we employ a cycle-consistency loss in the latent space to learn the symmetry transformation g. This method has various benefits in comparison to the adversarial approach. First, we overcome the alternating optimisation scheme which can result in severe convergence issues in practice (Mescheder et al., 2018). Second, we overcome the Gaussian assumption for minimising the mutual information which may results in model mismatch problems. Last, our method allows to deal with mixed continuous and discrete targets, simultaneously. Our results on both artificial and molecular datasets showcase the superiority of our approach in comparison to state-of-the-art methods.

7.2 Limitations

Extension to an Implicit Copula Formulation. In Chapter 4, we estimate the mutual information by employing an explicit copula transformation in a pre-processing step. However, this transformation defined by Eq. (4.2.6) requires ranking the data or learning the empirical cdf and then applying it to the data. That is, we require all data beforehand to perform the transformation. A potential approach for future work to overcome this limitation is to incorporate the copula transformation into the network architecture. This would allow to learn the empirical cdf end-to-end instead via a pre-processing step as proposed in Chapter 4. To do so, we would need to construct samples of the form $(x, CDF(x))$ first, which means that regions of the domain would have to be interpreted as dimensions. By applying the copula transformation directly to the data, we bypass the above considerations and reduce the complexity of the network while enriching the generality of the model.

Structured Symmetry Transformations. In Chapter 5 and 6, we proposed two approaches to learn symmetry transformations g if only point-wise samples from the invariant feature extractor f are known. To do so, we have learned a latent subspace that serves as a surrogate for g. However in many cases, the latent subspace is high-dimensional which makes it difficult to investigate this subspace. In order to overcome this limitation, a conceivable solution is to structure this subspace with the concept of archetypes (Keller et al., 2019). This allows a more targeted

exploration of the latent subspace which potentially leads to a better understanding of the symmetry transformation g, overall.

Regional Symmetry Transformations. In Chapter 5 and 6, we introduced two methods to learn global symmetry transformations g if only point-wise samples from the invariant feature extractor f are known. However, in multiple cases such symmetry transformation might only apply to a local part why it is unreasonable that we can learn a global symmetry transformation. In order to overcome this issue, we could learn multiple regional symmetry transformations which are specific to predefined regions of the data.

7.3 Outlook

We demonstrated that the proposed methods are capable of learning symmetry transformation for molecular data (Wieser et al., 2020) and are not merely limited to simple tabular datasets. For this reason, the introduced models are highly flexible and can be applied to various challenging problems in complex domains such as medicine or physics. Our goal is to investigate the fundamental questions in these domains by employing our presented methods and contribute to an improved understanding of the natural sciences.

References

Rdkit: Open-source cheminformatics, 2019. URL: `http://www.rdkit.org`.

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon & Kevin Murphy. Deep variational information bottleneck. *International Conference on Learning Representations*, 2017.

Nadine Bachmann, Chantal von Siebenthal, Valentina Vongrad, Teja Turk, Kathrin Neumann, Niko Beerenwinkel, Jasmina Bogojeska, Jacques Fellay, Volker Roth, Yik Lim Kok, Christian Thorball, Alessandro Borghesi, Sonali Parbhoo, Mario Wieser, Jürg Boni, Matthieu Perreau, Thomas Klimkait, Sabine Yerly, Manuel Battegay, Andri Rauch, Matthias Hoffmann, Enos Bernasconi, Matthias Cavassini, Roger Kouyos, Karin Metzner & Huldrych Günthard. Determinants of hiv-1 reservoir size and long-term dynamics during suppressive art. *Nature communications*, 10(1):1–11, 2019.

Diane Bouchacourt, Ryota Tomioka & Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *AAAI Conference on Artificial Intelligence,*.

Matthew Chalk, Olivier Marre & Gasper Tkacik. Relevant sparse codes with variational information bottleneck. In *Advances in Neural Information Processing Systems*. 2016.

Gal Chechik, Amir Globerson, Naftali Tishby & Yair Weiss. Information bottleneck for gaussian variables. In *Journal of Machine Learning Research*, 2005.

Thomas M. Cover & Joy A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2006.

Antonia Creswell, Anil A. Bharath & Biswa Sengupta. Conditional autoencoders with adversarial information factorization. *CoRR*, abs/1711.05175, 2017.

Mikhail Figurnov, Shakir Mohamed & Andriy Mnih. Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems 31*. 2018.

M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, Ö. Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski & D. J. Fox. Gaussian 09 Revision D.01. Gaussian Inc. Wallingford CT 2009.

Rafael Gomez-Bombarelli, Jennifer N. Wei, David Duvenaud, Jose Miguel Hernandez-Lobato, Benjamin Sanchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams & Alan Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville & Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*. 2014.

Shaobo Han, Xuejun Liao, David B Dunson & Lawrence Carin. Variational gaussian copula inference. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 2016.

Sepp Hochreiter & Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros & Trevor Darrell. CyCADA: Cycle-consistent adversarial domain adaptation. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Ayush Jaiswal, Daniel Moyer, Greg Ver Steeg, Wael AbdAlmageed & Premkumar Natarajan. Invariant representations through adversarial forgetting. In *AAAI Conference on Artificial Intelligence*. 2020.

Stewart Computational Chemistry James J. P. Stewart. MOPAC2016, 2016. URL: `OpenMOPAC.net`.

E. Jang, S. Gu & B. Poole. Categorical Reparameterization with Gumbel-Softmax. *International Conference on Learning Representations*, 2017.

Ananya Harsh Jha, Saket Anand, Maneesh Kumar Singh & V. S. R. Veeravasarapu. Disentangling factors of variation with cycle-consistent variational auto-encoders. In *European Conference on Computer Vision*, 2018.

Dinu Kaufmann, Sebastian Keller & Volker Roth. Copula archetypal analysis. In *German Conference on Pattern Recognition*, 2015.

Sebastian Mathias Keller, Maxim Samarin, Fabricio Arend Torres, Mario Wieser & Volker Roth. Learning extremal representations with deep archetypal analysis, 2020.

Sebastian Mathias Keller, Maxim Samarin, Mario Wieser & Volker Roth. Deep archetypal analysis. *German Conference on Pattern Recognition*, 2019.

Diederik P. Kingma & Jimmy. Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 2015.

Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende & Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems 27*, 2014.

Diederik P. Kingma & Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations*, 2014.

Jack Klys, Jake Snell & Richard Zemel. Learning latent subspaces in variational autoencoders. In *Advances in Neural Information Processing Systems 31*. 2018.

Adam Kortylewski, Aleksander Wieczorek, Mario Wieser, Clemens Blumer, Sonali Parbhoo, An-

dreas Morel-Forster, Volker Roth & Thomas Vetter. Greedy structure learning of hierarchical compositional models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Adam Kortylewski, Mario Wieser, Andreas Morel-Forster, Aleksander Wieczorek, Sonali Parbhoo, Volker Roth & Thomas Vetter. Informed mcmc with bayesian neural networks for facial image analysis. *arXiv preprint arXiv:1811.07969*, 2018.

Alexander Kraskov, Harald Stögbauer & Peter Grassberger. Estimating mutual information. In *Physical review. E, Statistical, nonlinear, and soft matter physics*. 2004.

Matt J. Kusner, Brooks Paige & José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International Conference on Machine Learning*, 2017.

Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer & Marc'Aurelio Ranzato. Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems 30*, 2017.

Ask Hjorth Larsen, Jens Jørgen Mortensen, Jakob Blomqvist, Ivano E Castelli, Rune Christensen, Marcin Dułak, Jesper Friis, Michael N Groves, Bjørk Hammer, Cory Hargus, Eric D Hermes, Paul C Jennings, Peter Bjerre Jensen, James Kermode, John R Kitchin, Esben Leonhard Kolsbjerg, Joseph Kubal, Kristen Kaasbjerg, Steen Lysgaard, Jón Bergmann Maronsson, Tristan Maxson, Thomas Olsen, Lars Pastewka, Andrew Peterson, Carsten Rostgaard, Jakob Schiøtz, Ole Schütt, Mikkel Strange, Kristian S Thygesen, Tejs Vegge, Lasse Vilhelmsen, Michael Walter, Zhenhua Zeng & Karsten W Jacobsen. The atomic simulation environment—a python library for working with atoms. *Journal of Physics: Condensed Matter*, 2017.

Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh & Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *The European Conference on Computer Vision*, 2018.

Ming-Yu Liu, Thomas Breuel & Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, 2017.

Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen & Jan Kautz. Few-shot unsupervised image-to-image translation. In *International Conference on Computer Vision*, 2019.

Ying Liu. *Directed Information for Complex Network Analysis from Multivariate Time Series*. PhD thesis, 2012.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling & Richard S. Zemel. The variational fair autoencoder. In *International Conference on Learning Representations*, 2016.

Jian Ma & Zengqi Sun. Mutual information is copula entropy. *Tsinghua Science & Technology*, 2011.

David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. 2003.

A. Mathur, A. Isopoussu, F. Kawsar, N. B. Berthouze & N. D. Lane. Flexadapt: Flexible cycle-consistent adversarial domain adaptation. In *International Conference On Machine Learning And Applications*, 2019.

Lars Mescheder, Andreas Geiger & Sebastian Nowozin. Which training methods for gans do actually converge? In *International Conference on Machine learning*, 2018.

Daniel Moyer, Shuyang Gao, Rob Brekelmans, Aram Galstyan & Greg Ver Steeg. Invariant representations without adversarial training. In *Advances in Neural Information Processing Systems*. 2018.

Roger B. Nelsen. *An Introduction to Copulas*. 2010.

Noel O'Boyle & Andrew Dalke. DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of Chemical Structures. 2018.

Sonali Parbhoo, Mario Wieser, Volker Roth & Finale Doshi-Velez. Transfer Learning from Well-Curated to Less-Resourced Populations with HIV. *Machine Learning for Healthcare*, 2020.

Sonali Parbhoo, Mario Wieser, Aleksander Wieczorek & Volker Roth. Information Bottleneck For Estimating Treatment Effects with Systematically Missing Covariates. *Entropy*, 2020.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser & Geoffrey E. Hinton. Regularizing neural networks by penalizing confident output distributions. *International Conference on Learning Representations*, 2017.

Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp & O Anatole von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 2014.

Melanie Rey. *Copula models in machine learning*. PhD thesis, 2015.

Mélanie Rey & Volker Roth. Meta-gaussian information bottleneck. In *Advances in Neural Information Processing Systems*, 2012.

Mélanie Rey, Volker Roth & Thomas Fuchs. Sparse meta-gaussian information bottleneck. In *International Conference on Machine Learning*, 2014.

Danilo Jimenez Rezende, Shakir Mohamed & Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

Thomas Robert, Nicolas Thome & Matthieu Cord. Dualdis: Dual-branch disentangling with adversarial learning, 2019.

Lars Ruddigkeit, Ruud van Deursen, Lorenz C. Blum & Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of Chemical Information and Modeling*, 2012.

W. Shockley & H. J. Queisser. Detailed Balance Limit of Efficiency of p-n Junction Solar Cells. *Journal of Applied Physics*, 1961.

Abe Sklar. *Fonctions de Répartition À N Dimensions Et Leurs Marges*. 1959.

Noam Slonim, Nir Friedman & Naftali Tishby. Agglomerative multivariate information bottleneck. In *Advances in Neural Information Processing Systems 14*. 2002.

Kihyuk Sohn, Honglak Lee & Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems*. 2015.

Teague Sterling & John J. Irwin. Zinc 15 – ligand discovery for everyone. In *Journal of Chemical Information and Modeling*. 2015.

S. Suh & S. Choi. Gaussian Copula Variational Autoencoders for Mixed Data. *ArXiv e-prints*, April 2016.

Christian W. Thorball, Alessandro Borghesi, Nadine Bachmann, Chantal von Siebenthal, Valentina Vongrad, Teja Turk, Kathrin Neumann, Niko Beerenwinkel, Jasmina Bogojeska, Volker Roth, Yik Lim Kok, Sonali Parbhoo, Mario Wieser, Jürg Boni, Matthieu Perreau, Thomas Klimkait, Sabine Yerly, Manuel Battegay, Andri Rauch, Patrick Schmid, Enos Bernasconi, Matthias Cavassini, Roger D. Kouyos, Huldrych F. Günthard, Karin J. Metzner, Jacques Fellay & the Swiss HIV Cohort Study. Host genomics of the hiv-1 reservoir size and its decay rate during suppressive antiretroviral treatment. *JAIDS Journal of Acquired Immune Deficiency Syndromes*, pages 517–524, 2020.

Naftali Tishby, Fernando C. Pereira & William Bialek. The information bottleneck method. pages 368–377, 1999.

Dustin Tran, David Blei & Edo M Airoldi. Copula variational inference. In *Advances in Neural Information Processing Systems 28*. 2015.

Aleksander Wieczorek & Volker Roth. Causal compression. *arXiv preprint arXiv:1611.00261*, 2016.

Aleksander Wieczorek & Volker Roth. On the difference between the information bottleneck and the deep information bottleneck. In *Entropy*, 2020.

Aleksander Wieczorek, Mario Wieser, Damian Murezzan & Volker Roth. Learning Sparse Latent Representations with the Deep Copula Information Bottleneck. *International Conference on Learning Representations (ICLR)*, 2018.

Mario Wieser, Sonali Parbhoo, Aleksander Wieczorek & Volker Roth. Inverse learning of symmetries. In *Advances in Neural Information Processing Systems*. 2020.

Qizhe Xie, Zihang Dai, Yulun Du, Eduard Hovy & Graham Neubig. Controllable invariance through adversarial feature learning. In *Advances in Neural Information Processing Systems*. 2017.

J. Zhu, T. Park, P. Isola & A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision*, 2017.

Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang & Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*.